THE ECONOMICS OF EXPERIMENTATION IN THE DESIGN OF NEW PRODUCTS AND PROCESSES

by

Stefan H. Thomke

B.S. in Electrical Engineering, University of Oklahoma (1988)
M.S. in Electrical Engineering, Arizona State University (1990)
S.M. in Operations Research, MIT (1993)
S.M. in Management, MIT (1993)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY
in Electrical Engineering and Management

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1995

Signature of Author.....
Massachusetts Institute of Technology
December 8, 1994

Certified by.....
Eric A. von Hippel
Chairman, Interdepartmental Doctoral Committee
Thesis Supervisor

Accepted by.....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Studies

THE ECONOMICS OF EXPERIMENTATION IN THE DESIGN OF NEW PRODUCTS AND PROCESSES

by

Stefan H. Thomke

Submitted to the Department of Electrical Engineering and Computer Science
on December 8, 1994, in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Abstract

Experimentation is a major innovation activity and accounts for a significant part of total innovation cost and time. In many fields, the economics of experimentation is being radically affected by the use of new and greatly improved versions of methods such as computer simulation, mass screening, and rapid prototyping. In this thesis, I show that a given experiment (and the related trial and error learning) can be conducted in different "modes" (e.g. computer simulation) and that users will find it economical to switch between these modes so as to reduce total design cost and time. These results have significant implications for research into problem-solving and the management of the innovation process.

In the thesis, I develop a (1) general model of experimental trial and error that regards experimentation as an iterative four-step *design-build-run-analyze* cycle; and (2) a decision model that aids in the evaluation and selection of experimentation modes prior to the start of an experimental cycle.

The following implications are derived from the general model:

— The efficiency of experimentation modes can vary with respect to conducting the four experimental steps within an experimental cycle.

— The efficiencies of repeated experimental cycles in selected modes can decrease at different rates.

— As a design progresses and a mode's efficiency decreases, there may be an optimal switching point (OSP) where switching to another mode is an economic strategy.

Furthermore, changes in experimentation model accuracy and/or model build cost shifts the OSP, and designers will find that adjusting their mode switching strategies in the direction of the new OSP will result in significant reductions of total design cost and time.

The general model and its implications are supported by the results of two field studies (one based on site visits and face-to-face interviews, and the other based on a large-scale survey) that compare computer simulation and prototype testing in the design of custom circuits containing electrically-programmable logic devices (EPLDs) and application-specific integrated circuits (ASICs).

The implications for managerial practice and innovation theory are discussed and suggestions for further research undertakings are provided.

THESIS SUPERVISOR: Eric A. von Hippel
TITLE: Professor of Management

Table of Contents

CHAPTER III

CHAPTER IV

CHAPTER V

CHAPTER VI

Acknowledgements

This thesis couldn't have been written without the help and support of many individuals. The members of my thesis committee — Don Clausing, Steven Eppinger, and Eric von Hippel — have given me valuable feedback at various points of my research and patiently reviewed an earlier draft of this thesis. I would like to express my thanks for their tolerance, friendliness, and intellectual support.

During the course of the last two years, I have talked to and corresponded with hundreds of people, and I would like to thank them for their help. Space does not allow me to thank individually all of those who took great time and trouble to participate in my field research, but I would like to thank particularly Phil Carvey, Larry Dennison, Imran Malik, and Ron Pettyjohn.

A special word of thanks goes to my thesis advisor, mentor, and teacher Eric von Hippel. I have spent the last two and a half years under Eric's guidance and these years have been the richest in my intellectual development so far. Eric has taken a great deal of time to teach me many things and through his friendship, he has made my stay at MIT an extremely enjoyable experience.

The individual that I would like to thank the most is my wonderful wife Savita who has patiently supported my academic journey for many years. This thesis would not have been possible without her love and support.

I would also like to thank the Lemelson-MIT doctoral fellowship program for generously supporting my research and education at MIT.

Dedication

मेरे प्यारे विक्रम के लिये

Chapter I

Introduction

"One should always keep things as simple as possible; but no simpler than that." **Albert Einstein**

I.1 Introduction

Today the efficiency of experimentation is being radically affected in many fields due to the use of new or greatly improved versions of methods such as computer simulation, mass screening, and rapid prototyping (these methods will be referred to as "experimentation modes"). In this thesis, I will examine the relative economies of these evolving forms of experimentation and study their impact on experimentation strategies in innovation.

Although, as will be seen in my literature review, much prior work exists that will help in my explorations, I know of no previous study that has investigated the micro-level mechanism of experimentation from an economic perspective in general, or that has investigated the economic consequences of new or improved experimentation modes for the design of new products and processes in particular.

With the aid of two models and two empirical studies, I will also introduce the concept of an "experimentation mode switching strategy" as a novel design management concept that can, if properly applied, result in significant reductions of total design cost and time[1]. Empirical findings show that such an experimentation strategy can sometimes benefit from "getting it [the prototype] *wrong* the first time"; a strategy that contradicts prior work in product development which prescribes "getting it *right* the first time" as a universal recipe for improved development performance.

I.2 Background on Research Problem

Experimentation, a form of problem-solving, is a major innovation process activity. Research into problem-solving shows it to consist of trial and error, directed by some amount of insight as to the direction in which a solution might lie (Baron 1988). This finding is supported by empirical studies of problem-solving in the specific arena of product and process development (Marples 1961, Allen 1966). Such studies show trial and error

[1] The economics of switching between experimentation modes may also depend on other factors that are not part of design development (e.g. production). While I discuss these factors at various points of my thesis, I will not explicitly include them in my empirical analysis.

(or, more precisely, trial, failure, learning, revision and re-trial) as a prominent feature (see also: Middendorf 1986; Newell and Simon 1972; Popper 1972; Simon 1981; Wheelright and Clark 1992).

The execution of an experiment can be seen as involving a four-step cycle:

(1) Design: One conceives of or designs an experiment.

(2) Build: One builds the (physical or virtual) apparatus needed to conduct that experiment.

(3) Run: One runs the experiment.

(4) Analyze: One analyzes the result.

For example, one might (1) conceive of and design a new, more rapidly-deploying airbag for a car; (2) build a prototype of key elements of that airbag as well as any special apparatus needed to test its speed of deployment; (3) run the experiment to determine actual deployment speed; and (4) analyze the result. (If the results of a first experiment are satisfactory, one stops. However, experimentation is usually a matter of repeated trial and error: That is, if analysis shows that the results of an experiment are not satisfactory, one modifies one's design on the basis of what one has learned and "iterates" or tries again.)

Each of the four steps of an experiment cycle has an efficiency[2]. The absolute and relative magnitude of these efficiencies can affect experimentation strategies. Thus, if experimenters have a choice of

---

[2] I define efficiency as the experimental "yield" per unit cost (where cost includes the cost of time). Experimental "yield" can be broadly defined as the value of new information that is "learned" during experimentation. An example would be the number of errors eliminated (the experimental yield) per experimental cycle, divided by the cost of conducting the cycle.

experimentation modes available (e.g. computer simulation and prototyping), they might logically switch between modes as to minimize the expected innovation cost.

Recently, new experimentation modes have been emerging which can radically affect the absolute and relative efficiencies of the four steps involved in experimentation. I will illustrate this by reference to three such modes: computer simulation, mass screening, and rapid prototyping.

Computer simulation is used as a substitute for "real experimentation" in fields ranging from the design of drugs (e.g., rational drug design) to the design of mechanical products (e.g., finite element analysis), to the design of electronic products (e.g., computer simulation of digital circuitry) to the analysis of global warming (e.g., climate modeling). An experimenter typically uses simulation in steps (3) and (4) of an experimental cycle — running an experiment and analyzing the result[3]. The ability to usefully substitute a simulation for a "real" experiment requires, of course, a simulation model that is accurate from the point of view of a given experimentation purpose.

Simulation can have both, disadvantages and advantages relative to testing a physical prototype. As an illustration, consider a real explosion in a prototype cylinder of a gasoline-powered car engine which may take milliseconds, while a detailed and less accurate simulation of that same

---

[3] Often experimenters are also required to invest resources in step (2) — building or completing a mathematical model to be used in simulation. For example, during the simulation of an integrated circuit, designers often have to enter their design in either a schematic or a language format, and develop software that describes the behavior of the system external to the circuit.

explosion might take minutes or hours on a powerful computer. On the other hand, the analysis of an experiment carried out in simulation is typically much richer. For example, one can only collect data from combustion in a car engine cylinder at a few points via instrument probes. But one can obtain information from a simulated explosion on variables such as gas temperature and shock wave propagation at any location and at any point in the evolution of the explosion. This allows experimenters to get much more and much better data per experimentation run. Thus subsequent experiments can be designed to be more efficient.

Mass screening is a very efficient way to achieve step (4) of an experimental cycle, that is the analysis (or, at least, identification) of experimentation results. In this method, one may screen the results of literally tens of thousands of experiments of a suitable type extremely rapidly and cheaply. For example, one can economically and simultaneously screen many bacteria for a gene giving resistance to a given antibiotic by placing a mixture consisting of millions of different types on an agar plate, treating the plate with the antibiotic in question, and then simply observing which of the many bacteria grow into colonies despite the treatment. Those that do grow in the presence of the antibiotic have the type of resistance one is seeking. Mass screenings of this general type are applicable to both unknown and known inputs. An example of an unknown input is when genetic mutations of many unknown types are created in bacteria by subjecting them to x-ray irradiation. Known inputs for mass screening can be created by, for example, techniques that systematically create and "tag" thousands of known

combinations of simpler chemicals (Amato 1992, Bergman and Gittens 1985, Stryer 1981).

Rapid prototyping is used by developers to quickly generate an inexpensive, easy to modify (and often physical) prototype that can be tested against the actual use environment and allows "real"[4] experimentation. Rapid prototyping techniques can be found in areas ranging from mechanical designs (e.g. stereolithography, three-dimensional printing) to the design of integrated circuits (e.g. electrically-programmable logic devices (EPLDs)[5]), to the design of software (e.g. emulation of user-interfaces) (Boehm, Gray and Seewaldt 1984; Carey and Mason 1986; Dvorak and Teschler 1993; Griffiths 1993; Mullins 1990; von Hippel 1994a; Wohlers 1993). The utilization of such rapid prototyping techniques has resulted in dramatic improvements in development time and cost. With increased competition and shorter product life cycles, accelerated time-to-market and lower development cost has become increasingly important to product success (Clark and Fujimoto 1991; Clausing 1993; Reinertsen 1992[6]; Smith and Reinertsen 1991; Wheelright and Clark 1992).

---

[4] Real is an abbreviation for "very accurate with respect to the final physical design and its intended operating environment".

[5] The technical literature sometimes refers to them as field-programmable logic devices or FPLDs.

[6] Using a simple economic model of new-product development and commercialization, Reinertsen (1983) finds that:
— In both high- and low-growth markets, overrunning development cost by 50% has less than a 4% impact on profits before tax.
— In a high growth market with short product life cycles, shipping a product six months late can draw down its life cycle profits by 33%. But in a slow-growth market with long product-life cycles, late shipment creates only a 7% decline in these profits.

Rapid prototyping is usually an inexpensive way to achieve step (2) in an experimental cycle while preserving the advantages of "real" experimentation — higher degrees of accuracy and experimentation speed (step (3)) with respect to the given experiment.

I.3 General Hypothesis

The general hypothesis is that new types of experimentation modes such as the three just described can have a great impact on experimentation strategies used in innovation. I propose that these new modes each have different impacts on experimentation efficiency and, as a result, users can benefit significantly from switching between experimentation modes as exploit these variations in mode efficiency. More specifically, I hypothesize the following:

— Experimentation modes (e.g. simulation, prototyping) vary with respect to efficiency of experimentation. As a design progresses, users will find it economical to switch between experimentation modes so as to reduce total design cost.

In other words, a designer of automobiles may find computer-based crash simulations to be of moderate accuracy and speed relative to crashing "real" prototypes. Nonetheless, computer simulations allow the designer to get more detailed data than a real crash would and that at a much lower cost per crash. But as a design progresses, it will be increasingly difficult (and

---

— In many cases, the single most important factor is cost. A product-cost overrun of only 9% results in a devastating 45% decrease in profits in a slow-growth market, while a similar overrun in a fast-growth market reduces profits by as much as 22%.

costly) to improve the accuracy of simulation by running more test cases and developing more accurate simulation models. Thus, at some point, designers are likely to switch to prototype testing since the incremental benefits of high model accuracy and fast experimentation speed exceed the incremental cost of using "real" prototypes (and the disadvantage of getting less detailed crash-related information). I propose that the optimal selection of this switching point (i.e. from simulation to prototype testing, and back) can have a significant impact on total design cost and therefore should be explicitly managed. Thus, in cases where the net benefit of prototype testing exceeds the net benefit of simulation, it does not make economic sense to continue with simulation in order "to get the prototype right the first time".

I.4 Examples of Fields That Are Radically Affected

In this thesis I will explore the issues just raised in a field that has been and continues to be dramatically affected by newly evolving forms of experimentation — the design of integrated circuit-based systems. In this section, I will give the reader (1) an initial overview of how experimentation in circuit design has been affected by radical improvements in simulation tools and prototyping technologies; and (2) demonstrate the general applicability of my research by describing a second area ("the design of pharmaceutical drugs") that is currently experiencing dramatic changes in the economics of experimentation.

I.4.1 The Design of Integrated Circuit Based Systems

To illustrate my general hypothesis, consider the effect that simulation and rapid prototyping has had on the economics of designing integrated circuit-based hardware systems used in custom applications such as data communications equipment, computers, etc.

Traditional hardware design has involved (1) the translation of design requirements into a candidate design solution (in the form of professionally drawn circuit diagrams); (2) the testing and modification of design solutions with the aid of "paper-based" simulations; and (3) further testing and modification of design solutions with physical hardware prototypes. During "paper-based" simulation, designers would verify the functionality of their conceived solution by drawing simulated waveforms for system input and output signals on paper. The efficiency of these simulations was highly contingent upon the designer's skill in selecting a small sample of test cases that reflected the most critical features of the design, and in accurately modeling waveforms as they propagated through the system to be designed. Because of limitations in speed (few test cases) and model accuracy (designers often use mental models) of simulating on paper, much iterative experimentation had to be performed with the aid of physical circuit prototype boards that could be modified at low-cost. (These boards are known as wire-wrapped breadboards. On these breadboards, one could set-up an arbitrary configuration of circuit components and connect them via physical wires that were wrapped around connection posts. If one wanted to make changes, the wires were simply re-wrapped at another post.)

With increases in design complexity and levels of integration, the traditional method of experimentation with prototypes became very inefficient. For example, the emergence of high density application-specific integrated circuits (ASICs) has enabled designers to integrate large systems into a single integrated circuit — an approach that has remarkable advantages with respect to increased performance and lower production cost (Walker 1992). However, from the perspective of design and experimentation, ASIC prototypes cannot be easily modified. In fact, a minor change to an ASIC prototype can amount to thousands of dollars in refabrication costs. As a result, the cost of experimentation increased as more effort had to be invested into "getting it right the first time" in prototyping; a goal that was difficult to achieve because of the described weaknesses of paper simulations.

Two developments have dramatically affected the traditional way of experimentation in circuit design. The first development, *computer simulation*, has enabled designers to simulate complex circuit behavior with both, a much higher level of speed and accuracy than traditional paper simulations. In fact, most designers would quickly agree that simulation tools have significantly lowered their total design cost and time. Today one typically prepares the design with the aid of computer-aided design (CAD) tools which, in turn, prepare files that can be used by simulation software. The second and most recent development, *rapid prototyping* with the aid of electrically-programmable logic devices (EPLDs), has enabled designers to quickly generate a low-cost hardware prototype of a proposed design solution and test it under real operating conditions (Bhatia 1994; Jones 1992; Tally 1990).

Computer simulation and EPLDs appear to have different effects on the efficiency of conducting the four experimental steps (design, build, run, and analyze) in traditional circuit design:

— Computer simulations run much faster (step (3) in an experimental cycle) than paper-based simulations which are limited by human information processing capabilities. In addition, some families of test cases can be generated by computer software which significantly reduces the time to set-up a test case (experimental step (2)).

— Computer simulations use component models that are more accurate than the mental models used in paper simulation. On the other hand, computer simulations are less accurate than physical prototypes that can be tested against real conditions. The difference in accuracy affects the efficiency of an experimental cycle.

— Computer simulations provide more efficient access to information than prototypes or paper-based simulation, affecting the efficiency of step (4) (analysis) in an experimental cycle. For example, physical integrated circuits consist of thousands of internal components but one cannot arbitrarily access internal nodes unless the device (or the design) is physically modified (and often permanently damaged). Such procedures are usually very costly and not always possible. In contrast, computer simulations can easily access internal nodes under arbitrary operating conditions since circuits are usually modeled at the component level.

— Rapid prototyping (EPLDs) has dramatically reduced the cost of step (2) (build) in the iterative experimentation with physical prototypes.

> While the cost of modifying an ASIC prototype can cost thousands of dollars and may take several weeks, the cost of reprogramming EPLDs is negligible for many device technologies[7] and the devices can be customized within a very short time[8]. Since prototypes can generally execute test cases much faster and more accurately than computer simulations, the iterative use of prototype testing has [again] become an economical experimentation strategy in circuit design[9].

I.4.2 The Design of Pharmaceutical Drugs

To further illustrate my general hypothesis with a different example, consider the different effects that rational (or structural-based) drug design and new mass (or random) screening techniques can have on the traditional economics of experimentation in the pharmaceutical industry.

(To make this example clear, the reader may find a little more information on the specifics of "rational drug design" to be useful. Rational drug design is a method in which knowledge of the molecular biology of organisms is combined with an ability to simulate the interaction of molecules in a computer. The goal of rational drug design is to actually

---

[7] For example, SRAM-based EPLDs can be erased and reprogrammed. Other technologies may require a new EPLD but their cost is still negligible compared to ASICs. More information can be found in appendix B ("Notes on the Design of Custom Circuits").

[8] The comparison is for low-density ASICs since current EPLD densities rarely exceed 20,000 gates.

[9] Judging from various interviews with hardware designers and information found in industrial journals, it seems that EPLDs have significantly reduced total development time, if compared to similar projects using ASICs. A test of this informal observation will be part of the second field study presented in chapter VI.

create and model on a computer molecules that will have a given biological effect. For example, one might now know enough about the ways that antibiotics interfere with the biology of given types of bacteria to create a new molecule from scratch that would interfere in a known way at a known point in a given bacterium's reproductive cycle. In rational drug design, one would create this molecule with the aid of computer simulations that would allow one to visualize the molecule being designed, and the interactions of this molecule with target molecules in the bacterium at issue. The structure and function of a specific molecule is initially unknown and has to be determined through sophisticated and costly analysis techniques such as X-ray crystallography, two dimensional nuclear magnetic resonance, or site specific mutagenesis.)

The traditional early research steps in the development of a drug have involved screening candidate chemical compounds to see if they have some biological effect. Thus, the time-honored method of discovering new antibiotics has been to, first, collect samples of dirt and other materials likely to harbor molds and other potential producers of compounds with an antibiotic effect. Next, extracts from these samples are screened by observing their effects on bacteria in laboratory tests. Extracts found to have some potentially useful antibiotic properties are then studied further to isolate and analyze the active molecules in the extract and to assess their potential utility as drugs. (Further steps in drug development involve preclinical and clinical trials, seeking FDA approval etc..)

Rational drug design and mass screening appear to have different effects on the traditional steps and costs of this traditional experimental method as follows:

— Rational drug design costs more than traditional methods in the experimental design step, because one is actually designing the material to be tested. On the other hand, when rational drug design works, it greatly reduces the cost of the testing and analysis steps because fewer compounds must be tested to find a successful one, and one knows the identity of the compound being tested at the start.

— Mass screening costs much less in the set-up stage than rational drug design. As soon as an efficient screen is set-up, large substance libraries (250,000 - 500,000 compounds) are screened to test them for biological activity. Thus it costs less to run the experiment per compound tested.

On the basis of the above information, one can reasonably hypothesize that, under conditions where rational drug design is possible, it will pay when the cost of screening (running an experiment and analyzing the results) is high. In contrast, when mass screening is possible, it will be most cost-effective when the cost of designing an experiment (as opposed to simply creating a mass of compounds for testing) is high. To illustrate this point, consider the recent development of combinatorial chemistry which is a methodology to generate large compound libraries (250,000 - 1,000,000 compounds) of very diverse molecules in matter of days or weeks. Combined with high efficiency random screening, combinatorial chemistry is currently revolutionizing the economics of experimentation in the design of new drugs.

In this regard, it is interesting to note that nature has typically taken the approach of creating many experiments (random mutation and sexual exchange of genes) and then screening in the successful ones (survival of the fittest). In contrast, human researchers have typically found it more economical to spend more time creating and testing a few, well thought out experiments in the expectation of a high success rate. Rational drug design represents an advance in the traditional experimentation pattern. In contrast, "design by mass screening" represents a shift towards strategies more like those used in nature.

I.5 Expected Contribution to Academic Literature and Industrial Practice

The findings of this thesis will benefit researchers and managerial practitioners in a number of ways:

— Academics will have the results of a first study on the economics of experimentation in the design of new products and processes, and they may find the subject to be a useful new element to explore. For example, researchers interested in product development will find that the new developments in experimentation techniques and their resulting experimentation strategies have a significant impact on overall development performance, as evidenced by improvements in the design of custom circuits and pharmaceutical drugs.

— Those interested in the economics of innovation in general will find my results useful as micro-level data on a rapidly-changing innovation cost component. Existing work in the economics of innovation often focuses on industry or firm-level parameters which prevents

researchers to gain a thorough understanding of how expectations about cost affect innovators' behaviors. My research has shown that experimentation cost accounts for a significant component of total innovation cost and I believe that understanding the economics of experimentation constitutes a major step towards understanding innovation cost dynamics.

— Most research on "learning" has focused on how increases in production volume has affected improvements in efficiency. More recently, researchers have started to investigate micro-level mechanisms of learning but many research questions are still open. This thesis should allow students of learning to develop a better understanding of learning in design and how the evolution of new experimentation modes has affected learning rates. In fact, my research indicates that new experimentation modes and the related mode switching strategies can result in "accelerated learning", which is reflected in a reduction of total design cost.

— Innovation practitioners and managers are now adopting and using some of the new experimentation modes that will be discussed as part of this thesis. However, they typically appear to adopt these without any explicit understanding of the change in experimentation economics that can follow, and the consequent change in economic innovation strategies that may result. The models and findings presented in this thesis should allow developers of new products and processes to make more cost-effective judgments with respect to the experimentation

strategies they elect to pursue and provide the first source of insight in this matter.

I.5 Research Flow and Thesis Structure

My overall research flow is shown in figure I.1(next page). and is also reflected in the thesis structure discussed below. In **chapter II**, I will review three bodies of literature that relate to my work: (1) the study of experimental trial and error in problem-solving in general; (2) the study and modeling of iterations in design; and (3) the study of learning processes in design and manufacturing.

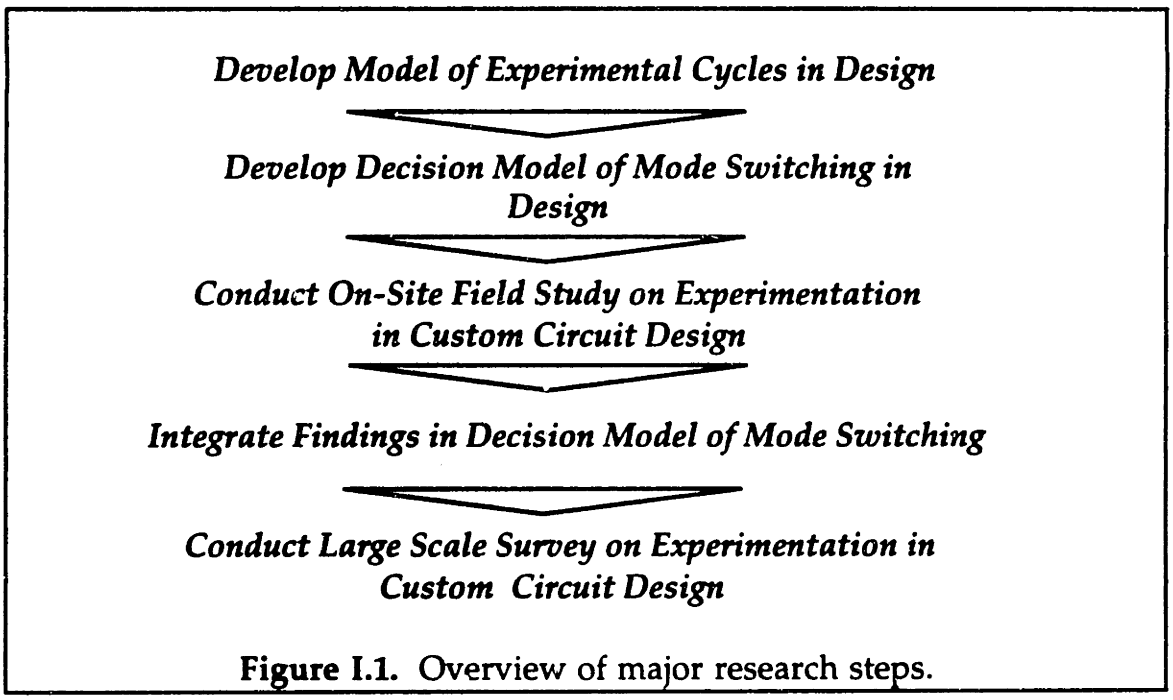


Figure I.1. Overview of major research steps.

In **chapter III**, I will develop a general four-step model of experimental cycles in design and use the model to develop a hypotheses structure that will be tested in the empirical part of the thesis. In **chapter IV**, I will use the

general model from chapter III to develop a decision model that focuses on the dynamic evaluation of experimentation modes in design. Then, **chapter V** reports the findings of the first [on-site] field study on experimentation strategies in circuit design and use these findings (1) to apply and to refine the decision model developed in chapter IV; and (2) to submit some preliminary support for the hypotheses structure that is developed chapter III. In my second field study (**chapter VI**), I rigorously test the general hypotheses structure with the aid of data from 391 circuit designers who participated in a large scale survey on experimentation strategies. Finally, **chapter VII** concludes the thesis with a summary of the research results and their implications for managerial practice and research. Suggestions for future research are provided.

Chapter II

Literature Review

II.1 Introduction

The importance of iterative experimentation to problem-solving and design has been referenced by a number of researchers. In this chapter, I review the following body of literature that directly relates to my thesis[1]: (1) the study of experimental trial and error in problem-solving in general; (2) the study and modeling of iterations in design; and (3) the study of learning

[1] Other relevant literature will be referenced (and summarized) throughout the thesis.

processes in design and manufacturing. My review of the literature results in the following findings:

— The general problem-solving literature recognizes experimental trial and error (and the related learning) as a fundamental problem-solving activity.

— Research into the design of new products identifies two types of iterations: (1) design iterations that are due to uncertainty of information that is outside (exogenous to) a design activity (e.g. information from customers or other design activities); and (2) design iterations that are due to learning inside (endogenous to) a design activity (e.g. experimental trial and error).

— Research into design iteration models and strategies has primarily focused on iterations of type (1). To the best of my knowledge, no research on the micro-level processes (and the related models and strategies) of experimental trial and error in design has been undertaken.

— Research into the micro-level mechanisms of "learning by doing" has only started very recently and significant opportunities to contribute to this area of research still exist.

— In chapter I, I observed that new and rapidly improving experimentation techniques such as computer simulation have dramatically changed the economics of experimental trial and error. Thus, there is an exciting opportunity to develop models of experimental trial and error in design, and to study the impact of new experimentation techniques on experimentation strategies in design.

II.2 Experimental Trial and Error and Iterations in Innovation

In this section, I will examine the literature and research (1) on experimental trial and error in problem-solving; (2) on iterations in the design of new products; (3) on evolutionary design in software engineering as an example of strategies to deal with iterations due to the uncertainty of exogenous information; and (4) on models of design iterations.

II.2.1 Experimental Trial and Error in Problem-Solving

Research shows that all problem-solving involves thinking and all thinking involves trial and error whenever one does not know immediately how to achieve a given goal[2] (Barron 1988). Thinking of a "possibility" becomes the trial and rejecting the "possibility" becomes the error. Empirical studies have shown that trial and error is not merely a random process but problem-solvers actually apply *insight* as to follow a more deterministic trial and error path. Thus Barron concludes that experimentation consists of trial and error, directed by some amount of insight as to the direction in which a solution might lie. This finding is supported by researchers of engineering design (Marples 1961, Allen 1966) who found trial and error (or, more precisely, trial, failure, learning, revision, and re-trial) as a prominent feature of technical problem-solving.

Interestingly, the importance of iterative trial and error as a fundamental process in advancing knowledge (or learning) can also be found in the research into the evolution of knowledge. Popper (1972) proposes that

---

[2] A number of empirical studies have shown that even animals use some degree of insight in performing iterative trial and error to solve problems (see Barron (1988), p. 44-45).

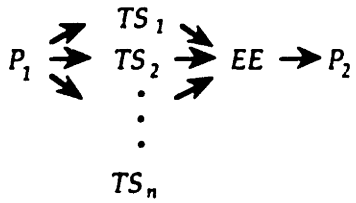
the activity of understanding can be represented as a general schema of problem-solving by the method of imaginative conjectures and criticisms (also known as the method of conjecture and refutation). He proposes the following schema (in its simplest form)[3] :

$$P_1 \rightarrow TT \rightarrow EE \rightarrow P_2$$

where P_1 is the starting problem, TT (the "tentative theory") is the imaginative conjectural solution which we first reach (for example our first tentative interpretation) and EE ("error elimination") consists of a critical examination of our conjecture. Our tentative interpretation consists, for example, of the critical use of available evidence and, if we have at this early stage more than one conjecture at our disposal, it will also consist of a critical discussion and comparative evaluation of the competing conjectures. Then P_2 is the problem as it emerges from our first critical attempt to solve our problems. It leads to our second attempt (and so on). Thus Popper concludes that "a satisfactory understanding will be reached if the interpretation, the conjectural theory, finds support in the fact that it can throw new light on new problems — on more problems than we expected; or if it finds support in the fact that it explains many subproblems, some of which were not seen to start

---

[3] In a latter essay, Popper noted that "...even in this form an important element is missing: the multiplicity of the tentative solutions, the multiplicity of the trials. Thus our final schema becomes something like this [TS stands for tentative solution]:



..."

with. Thus we may say that we can gauge the progress we have made by comparing P_1 with some of our later problems (say P_n)...."

II.2.2 Iterations in the Design of New Products

In this section, I analyze literature that includes references to the use of iterations in design. More specifically, I divide the literature in two sections: (1) references to design iterations that are due to uncertainty of information that is outside (exogenous to) a design activity (e.g. information from customers or other design activities); and (2) references to design iterations that are due to learning inside (endogenous to) a design activity (e.g. experimental trial and error). I will now discuss both sections in turn.

Design Iterations Due To Uncertainty of Exogenous Information

The use of iterative approaches to deal with uncertainty (or changes) of exogenous information has been referenced by a number of researchers. Thus, Suh (1990) describes that "...the design process begins the recognition of a societal need. The need is formalized, resulting in a set of functional requirements. The selection of functional requirements, which defines the design problem, is left to the designer. Once the need is formalized, ideas are generated to create a product (or an organizational structure). This product is then analyzed and compared with the original functional requirements through a feedback loop. When the product does not fully satisfy the specified functional requirements, then one must come up with a new idea, or change the functional requirements to reflect the original need more

accurately. This iterative process continues until the designer produces an acceptable result."

Von Hippel (1994a) proposes that in order to solve problems, information and problem-solving capabilities have to be brought together. But often the information used in technical problem-solving is costly to acquire, transfer, and use in a new locus — is in his terms "sticky". If and as each cycle requires access to sticky information located at more than one site, we can often observe iterative shifting of problem-solving activities between sticky information sites[4] (an iterative problem-solving pattern often encountered in new product and service development is depicted in figure II.1 on the next page).

Relating iteration and the incompleteness of design information, Ostrofsky (1977) observes that "... from the recognition of needs in the feasibility study to the last step in retiring the system a great deal of knowledge is gained concerning the system being developed. Unfortunately, decisions must be made at each step in the process in order to achieve the necessary conclusions, usually within existing time constraints. Because of the incompleteness of knowledge when decisions are made, they may require reexamination at subsequent points in time when additional information is gained. This process of reexamination is the iterative nature of design and is recognized as an integral part of the process."

---

[4] In order for shifting to occur, the cost of shifting intermediate outputs between sites has to be less than the cost of transferring the sticky information operated upon.

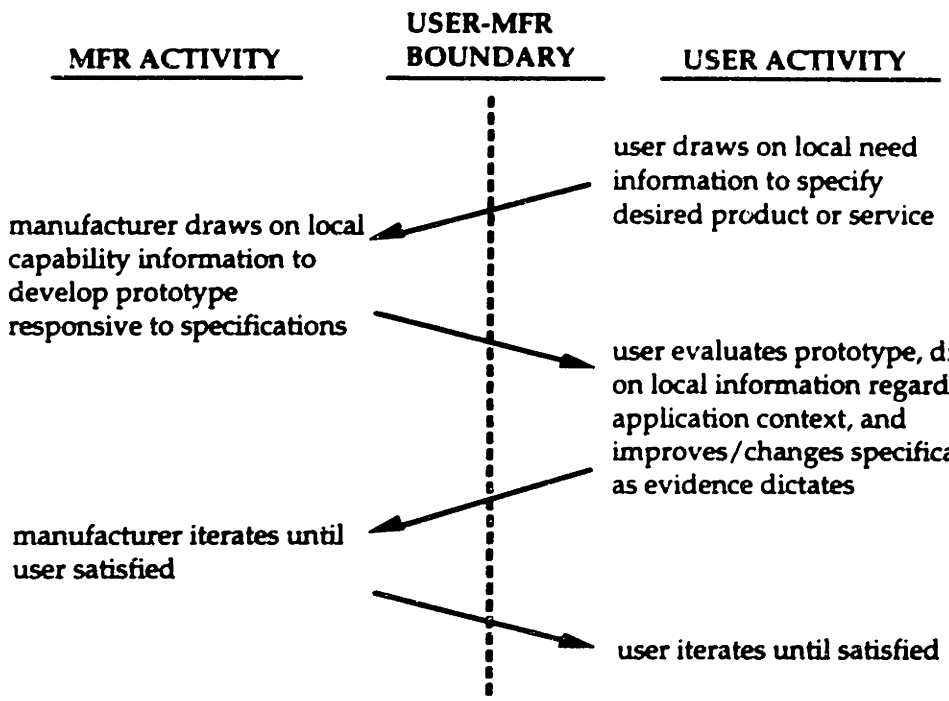


Figure II.1. A problem-solver may first draw on user need information to generate some attributes for a desired new product or service. Then, manufacturer information may be drawn upon in order to develop a prototype that appears responsive to the specification. The prototype is next tested within its proposed use context to verify function and accuracy of the initially stated need. If the two do match satisfactorily, the loci of need and/or capability information must be revisited in search of a closer match. This cycle may be repeated few or many times until an acceptable match is found. (Redrawn from von Hippel 1994a.)

Design Iterations Due To Endogenous Learning

The use of iterations (or experimental trial and error) as a means of integrating learning that is endogenous to a design activity has been referenced by a number of researchers. Middendorf (1986) distinguishes between three [experimentation] processes a designer can pursue in the detailed design of devices and systems. The first process, device evolution,

relies on drawings and physical models as no mathematical models are possible. Designers make a preliminary drawing, then a physical model, test it, intuitively decide upon changes which are thought to improve the product, and make a new model. This process is repeated until specifications are met. The second process, design by repeated analysis, utilizes analytical models (functional block diagrams, network models, and mathematical models) by assigning values to design parameters and iteratively evaluate specifications. The third process, synthesis, relies on a complete mathematical model to arrive at a solution with a single iteration. Middendorf emphasizes that such an approach is only possible if the relations are simple enough to permit an explicit solution.

Simon (1981) proposes that complex structures can be designed by discovering viable ways of decomposing it into semi-independent components corresponding to its many functional parts[5]. He suggests that "... one way of considering the decomposition, but acknowledging that interrelations among the components cannot be ignored, is to think of the design process as involving, first, the generation of alternatives and, then, the testing of these alternatives against a whole array of requirements and constraints. There need not be merely a single generate-test cycle, but there can be a whole nested series of such cycles...".[6]

Wheelright and Clark (1992) extend the idea of a generate-test cycle to product development by defining it as the "design-build-test" cycle (depicted

---

[5] Methods to partition tasks as to minimize interdependencies have been described by Alexander (1964) and von Hippel (1990).

[6] A similar approach has been first proposed by Marples (1961).

in figure II.2). They find that in practice, designers often follow such a cyclical (iterative) design approach.

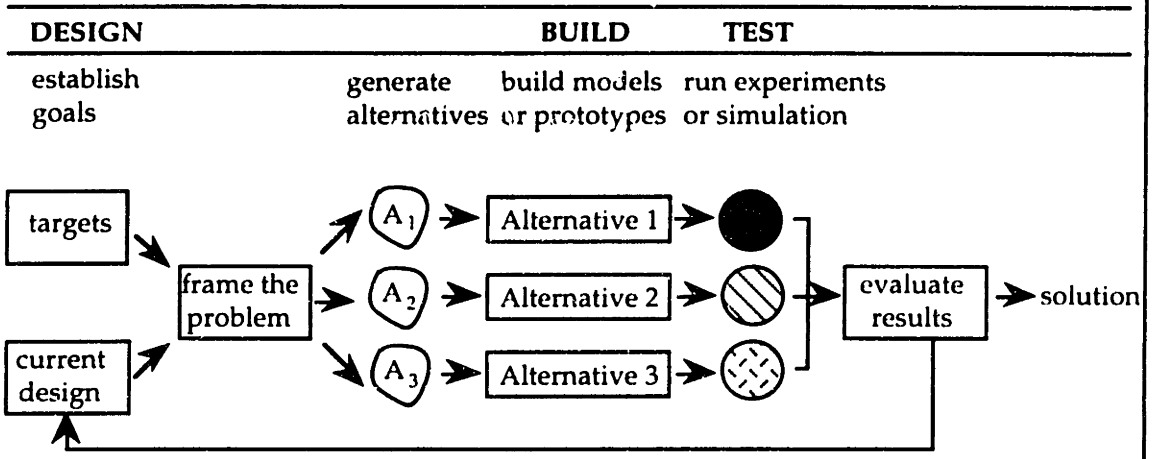


Figure II.2. The diagram illustrates the design-build-test cycle. Goals are set, problems are framed, alternatives developed, models or prototypes built, and tests conducted. If the test reveals a solution, the process stops. If no solution is found or the goals are not yet reached, the cycle is repeated. Note that repeating the cycle may involve revising the goals. Later cycles are often of shorter duration than earlier cycles. (Redrawn from Wheelright and Clark 1992).

II.2.3 Strategies to Deal with the Uncertainty of Exogenous Information: Evolutionary Design in Software Engineering

In the previous section, we have seen that design iterations can be divided into two types: (1) design iterations that are due to uncertainty of information that is outside (exogenous to) a design activity; and (2) references to design iterations that are due to learning inside (endogenous to) a design activity. In this section, I will briefly discuss the impact of evolutionary design approaches in software engineering — a design strategy that tries to address problems that are due to iterations of type (1). To the best of my

knowledge, no research has focused on the development of strategies that address iterations of type (2).

The application of evolutionary approaches to software engineering has received much attention over the last few years, particularly after some firms reported significant improvements in development performance that could be directly attributed to the use of evolutionary software design. The new approaches, known as "evolutionary systems development methods", can be best defined as: "An approach whereby the initial design proposals are put forward in the form of physical working model. The analysts, working in partnership with the users, gradually improve and develop this prototype of the system until it meets a level of acceptability decided on by the user. When this happens, the prototype becomes the new system." (Crinnion 1991).

Boehm *et al* (1984) report on a controlled experiment in which seven software teams developed the same small-size application software product using two different approaches[7], specifying (traditional method) and prototyping (evolutionary method). Their main findings are:

— Prototyping resulted in products with roughly equivalent performance but with 40 percent less code and 45 percent less effort.

— The prototypes rated somewhat lower on functionality and robustness, but higher on ease of use and ease of learning.

---

[7] Each method involved the following steps:

— Specifying:	Develop requirements specifications; develop design specifications that implement requirements; develop code to implement design; rework the result as necessary.
— Prototyping:	Build prototype version of parts of the product; exercise the prototype parts to determine how best to implement the operational product; proceed to build the operational product, and again rework as necessary.

— Specifying produced more coherent designs and software that was easier to integrate.

While Boehm *et al* demonstrate the advantages of using prototyping in their particular setting (applications software), they also note that prototyping may not always be preferable, especially in cases where the information value of a prototype may not be worth the investment in it.

The ability to rapidly build "software prototypes" has been particularly enhanced by the emergence of object-oriented programming systems (OOPS) (Mullins 1990). In the OOPS concept, programs are developed with the help of building blocks, where each block, termed an object, is independent and able to run by itself or can be interlocked with other objects. Entire programs can be developed by reusing existing objects and changing them without affecting other objects, as they are independent. Objects interact by passing information between one another. Programmers are only concerned about the function of each object without having to worry about its source code. While such an approach limits the degrees of freedom in software design, it also reduces design complexity and thus decreases prototyping cost. (The granularity of objects is the limiting variable in determining the degrees of freedom. If desired functions are not feasible with existing objects, one has to either create new objects or modify existing ones.) For example, an object that calculates the rates of return on an investment might link to an object that displays a graph of the results. The first object needs to know nothing about how the second plots the graph. Each object must merely contain information about itself ("encapsulation") and the object it can relate to ("inheritance") (Keen 1991).

Other developments in software engineering that try to overcome some of the shortcomings (e.g. poor modularity, difficulties in integration) of evolutionary approaches have been proposed. For example, Boehm (1988) proposes the usage of a "spiral" process model in developing software. In essence, the spiral model is a risk-driven design approach that conforms to the design problem in question. In cases where software requirements are well-known and stable (low redesign risk), the spiral model emphasizes traditional design approaches. In cases of high risk and requirement uncertainty, however, the model tends to emphasize a more prototype-driven evolutionary approach.

II.2.4 Models of Design Iterations

While most researchers acknowledge that iterations are a prominent feature of design, very few have actually tried to model design iterations mathematically and used these models to prescribe strategies that could improve design performance. Because models are also important in the development of a conceptual understanding of design iterations, I will now review three papers that have made important contributions to the modeling of task iterations in engineering design. The papers focus on the first type of iterations in design: task iterations that are due the incompleteness (or uncertainty) of needed information exogenous to a design task. To the best of my knowledge, no research has been done on modeling design iterations of the second type (i.e. iterations due to experimental trial and error that is endogenous to a design task).

The models discussed consider the impact of incomplete information on three different possibilities of design task completions[8]: (1) purely sequential task completion; (2) completely parallel task completion; and (3) overlapped task completion.

Design Iterations in Sequential Task Completion (Smith and Eppinger 1991)

In their research, Smith and Eppinger (1991) have developed a Markov chain-based model that describes the iterative solution progress in complex engineering design projects and that is used to reduce total design time by reordering the sequence of task completion[9]. In their model, a design project is divided into a series of sequential tasks which are completed one at a time. The model assumes that a task is an information processor, requiring input information from other tasks and, upon task completion, generating output information which will serve as input to other tasks[10] (i.e. tasks transform information). Furthermore, the model assumes that a task can be attempted with incomplete information but in such a case, there is a finite probability that the task will have to be repeated (task iterations). Thus tasks are

---

[8] The first two models utilize a tool known as the Design Structure Matrix (DSM). The original DSM uses a binary structure to represent the coupling between design tasks, and can be repartitioned as to minimize task iterations (Steward 1981).

[9] Cesiel (1993) proposes that, in addition to task reordering, task iterations can be further reduced by identifying system interdependencies using a quantitative method which is based on the statistical design of experiments. He suggests that design processes often have interdependencies in place that lead to more iterations than what is required by the system. In a case study of an automotive catalytic converter diagnostic calibration procedure, he finds that even after reordering tasks, iterations can be further reduced by modifying the design procedure along the interdependencies of system parameters.

[10] The model implicitly assumes that no information is exchanged during task execution.

dependent (uni- and bi-directionally) and the nature of dependence is a function of their mutual information needs. To illustrate the information dependence between tasks, consider two tasks A and B where A precedes B in task execution (the design structure matrix is shown in figure II.3).

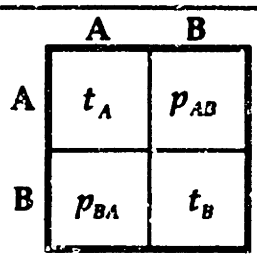


Figure II.3. Design structure matrix for two sequential tasks A and B. The expected task durations are shown as diagonal terms and the repeat probabilities are shown as off-diagonal terms.

I will first consider (1) the case where A and B are codependent (bi-directional information needs); and then (2) the case where only B depends on information from A (uni-directional information needs). So in the first case, a successful completion of A depends on information from B. But since B is executed after A, A will have to proceed without information from B and there is a finite probability that A will have to be repeated (iteration) once information from B becomes available. Since B depends on information from A, a repetition of A will also release new information to B, resulting in a repetition of B (at a finite probability), and so forth. The two tasks will iterate back and forth, with a rapidly decreasing expected task duration. In the second case, A will not be repeated since it does not require information from B. B will not be repeated since it merely requires information from A and A

is completed prior to B. Thus, the sequence A-B will not experience any iterations due to incomplete information.

Design Iterations in Parallel Task Completion (Smith and Eppinger 1992)

Similar to their work on sequential task iterations, Smith and Eppinger (1992) have developed an iteration model that can predict slow and rapid iteration within a project, and predict those features of a design problem which will require many iterations to reach a technical solution. The model assumes that a process can be divided in parallel tasks which exchange information at the beginning and the end of a task but not during task completion. Tasks can be executed with incomplete information (i.e. information from other tasks may not be available at the time of task execution) which leads to iterations once the information becomes available. In other words, if A and B are co-dependent and completed in the same stage (concurrently), output information from B will require A to iterate (rework), and vice versa. Rework is created based on the % of work done and thus task rework slowly declines to zero but at different rates of convergence for each task. Smith and Eppinger use a spectral decomposition of a work transformation model to find a dominant design mode which converges slowest and therefore represents a critical path for the time-progression towards a design solution.

Design Iterations in Overlapped Task Completion (Krishnan, Eppinger, and Whitney 1993)

As time-to-market is becoming more important in many industries, firms have resorted to a number of strategies to accelerate product

development, and the overlapping of design activities is one of them. In their model-based framework to overlap product development activities, Krishnan, Eppinger, and Whitney (1993) assume that the overlapped design tasks are coupled through the arbitrary exchange of information during task execution. But upstream and downstream tasks are coupled in a different way than the previous two models: downstream tasks depend on upstream information, but not vice versa. Downstream tasks can be completed with incomplete (or uncertain) upstream information but downstream iterations may be necessary to accommodate changes in upstream information. Thus, the likelihood of downstream iterations increases with higher degrees of overlapping as more upstream information is exchanged too early. With the aid of their model, Krishnan *et al* propose different overlapping strategies and derive the conditions under which these strategies are most effective.

II.2.5 Summary

In summary, the literature review has resulted in the following findings:

— The general problem-solving literature recognizes experimental trial and error (and the related learning) as a fundamental problem-solving activity.

— Research into the design of new products identifies two types of iterations: (1) design iterations that are due to uncertainty of information that is outside (exogenous to) a design activity (e.g. information from customers or other design activities); and (2) design

iterations that are due to learning inside (endogenous to) a design activity (e.g. experimental trial and error).

— Research on design iteration models and strategies has primarily focused on iterations of type (1). To the best of my knowledge, no research on the micro-level processes (and the related models and strategies) of experimental trial and error in design has been performed.

— In chapter I, I observed that new and rapidly improving experimentation techniques such as computer simulation have dramatically changed the economics of experimental trial and error. Thus, there is an exciting opportunity to develop models of experimental trial and error in design, and to study the impact of new experimentation techniques on experimentation strategies in design.

II.3 Learning by Experimentation

Another body of literature that relates to experimental trial and error is research often identified under the general heading of "learning by doing". As we have observed in the previous section, a fundamental component of experimental trial and error is the micro-level mechanism of *learning* but little research has been performed on such learning processes in the design of new products and processes.

Research into the systematic reduction of the unit cost of manufactured goods has shown the unit cost of producing manufactured goods to decline significantly as more are produced (Wright 1936) — a process that Arrow (1962) has termed as "learning by doing". Until recently most studies on learning have focused on better measures of the aggregate learning curve

effect, as evidenced by numerous efforts on finding appropriate proxies for "production experience" (Adler and Clark 1991). However, most of these experience measures (such as cumulative output) confound all micro-level learning mechanisms in a single variable and provide little insight on how learning occurs and how it can be effectively managed.

More recently, Rosenberg (1992) has shown that gains in productivity can also accrue by improvements in skill and understanding of a product design (which are not necessarily caused by production volume) — a process he termed "learning by using". (For example, after a given jet engine has been in use for a decade, the cost of maintenance may have declined to only 30% of the initial level as a result of "learning by using".)

In this section, I report on two recent papers that have made significant contributions to the understanding and management of micro-level learning in innovation but many opportunities for research still exist.

A Sketch of the Learning Process (Adler and Clark (1991))

Adler and Clark (1991) explore the behavioral processes that give rise to learning by doing in an intense study of eight departments of an electronic equipment firm. In their study, they focus on two different processes that drive overall learning:

— First-order learning: informal, tacit, and experiential learning-by-doing that fuels productivity directly and is based on repetition and on the associated incremental development of expertise (e.g. motor learning). First-order learning makes direct workers more effective in executing the tasks assigned to them.

— Second-order learning: formal, explicit, and cognitive learning-by-doing that fuels productivity indirectly by transforming the goals of a process by explicit managerial or engineering action to change the technology, the equipment, the process or the human capital in ways that augment capabilities.

Adler and Clark measure first-order learning by traditional expressions of experience — cumulative output and time. In contrast, second-order learning is measured by two types of learning activities: (1) changing the product/process design through engineering changes; and (2) building human capital through training. Since engineering activity and training are variables that can be directly managed, their influence on learning and productivity is of great importance to industrial practitioners.

Using cost accounting and industrial engineering reports from two departments, they construct empirical models of productivity (ratio of total output to total input) as a function of cumulative output volume, cumulative engineering activity, and cumulative training activity. Some of the findings that Adler and Clark report are:

— Second-order learning can disrupt and facilitate first-order learning which is quite surprising.

— A significant portion of second-order learning activity is prompted by problems that emerge during the accumulation of production experience.

— Second-order learning initiatives in one domain (e.g. engineering changes) can force changes in others (e.g. training) with cascading disruption effects.

As a result of their findings, Adler and Clark propose a learning process model which is shown in figure II.4. They conclude that the learning process is internally complex because, in order to respond to problems uncovered in the production process, some actions undertaken can have sizable negative effects on performance.

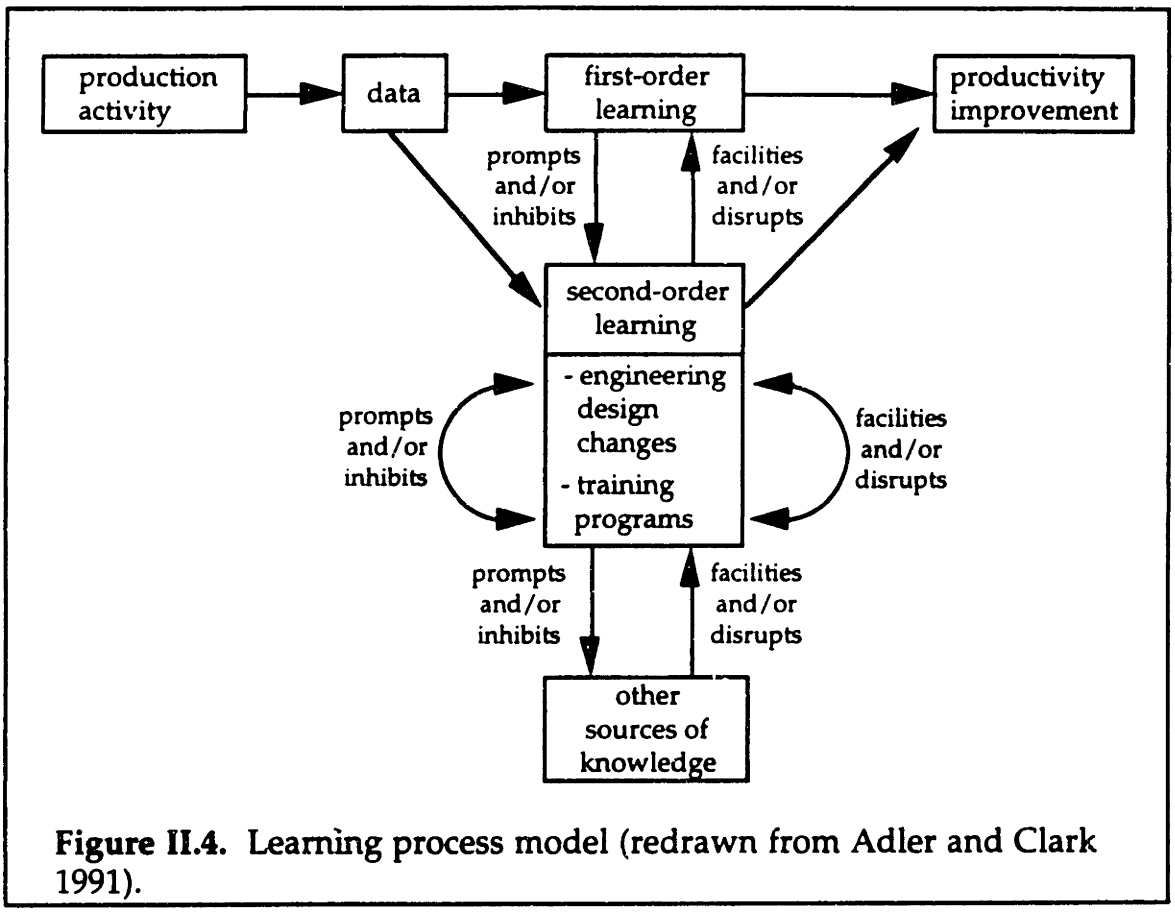


Figure II.4. Learning process model (redrawn from Adler and Clark 1991).

Adler and Clark's identification and measurement of second-order learning is an important contribution toward the understanding of learning by doing in general and the management of learning in production processes in particular. Unfortunately, it does not examine the micro-level mechanism by which learning by doing is actually done, or why doing is essential to

learning. However, a significant step toward the micro-level understanding of learning by doing can be found in von Hippel and Tyre (1994).

Learning by Doing and Interference Finding (von Hippel and Tyre (1994))

In their research on problem identification in novel process equipment, von Hippel and Tyre (1994) describe "interference finding" between an experimental object and its experimental context as the central mode of learning. Interference finding can be described as a form of pattern recognition. A pattern is essentially a set of features or characteristics that describes an object (or event, or stimulus). This bundle of features then may be used as a standard against which one may compare new objects. Thus one may wish to focus on the *similarities* between patterns in a process called pattern matching. (For example, systems designed to recognize objects with known characteristics ranging from handwriting to military targets often use algorithms based on pattern matching.) Or, one may wish to use subtractive pattern-matching to highlight the *difference* between two or more patterns. (For example, astronomers may compare two star maps of the same area of sky taken at two different times in order to subtract everything that is the same and highlight only what is changing — rapidly moving comets for example.)

Von Hippel and Tyre point out that interference finding is a form of pattern-matching which is sensitive to the deviations from the expected or the desired outcome pattern of an experiment. They quote Alexander (1964) who describes the essence of exception finding when he discusses a means for characterizing the fit between form and context:

> "It is common practice in engineering, if we wish to make a metal face perfectly smooth and level, to fit it against the surface of a standard steel block, which is level within finer limits than those we are aiming at, by inking the surface of this standard block and rubbing our metal face against the ink surface. If our metal face is not quite level, ink marks appear on it at those points which are higher than the rest. We grind away these high spots, and try to fit it against the block again. The face is level when it fits the block perfectly, so that there are no high spots which stand out any more."

Interference finding in most experiments is a more complex version of the process just described. Consider, for example, the process of discovering problems in the functioning of a novel process machine (the experimental object) by running it in a factory for the first time (the experimental context). Here, two very different and highly complex patterns, the new machine and the plant context, are brought in close juxtaposition during the first field use — the experiment. As a result, previously unsuspected and often subtle problems are discovered (the experimental learning in this case) because they evoke an obvious exception from expected machine performance. As an illustration, consider the following actual example of problem discovery via factory use of a novel process machine.

The Yellow Circuit Board Problem

> One of the novel process machines examined by von Hippel and Tyre was a "component placing machine", used in the process of assembling printed circuit boards. The task of this machine was to pick up electronic components from storage trays and place them at precise locations on printed circuit boards. The machine consisted of a robot arm coupled to a small

vision system. The vision system used a TV camera to locate specific metalized patterns on the surface of each circuit board being processed. For the vision system to function properly, this camera needed to be able to distinguish the metalized patterns on a circuit board clearly against the background color of the board surface itself.

The vision system developed by the machine development group functioned properly in the lab when tested with sample boards from the user plant. However, when it was introduced into the factory, users found that it sometimes failed, and called this to the attention of the machine developers. The development engineers came to the field to investigate, and found that the failures were occurring when boards that were light yellow in color were being processed — because the TV camera used could not "see" the metalized patterns on a circuit board against a light yellow background.

The fact that boards being processed were sometimes light yellow was a surprise to lab personnel. While the factory personnel knew that the boards they processed varied in color, they had not volunteered the information to the lab because they did not know that the designers would be interested. Early in the machine development process, factory personnel had simply provided samples of boards used in the factory to the lab. And, as it happened, these samples were green in color. On the basis of the samples, developers had then (implicitly) assumed that all boards processed in the field were green. It had not occurred to them to ask users, "how much variation in board color do you generally experience?" Thus, they had designed the vision system to work successfully with boards that were green.

In this case we can see that the learning resulting from the first field use, "the experiment", was the identification of an interference of machine

performance and the field. This interference was then traced by a diagnostic process to a previously unrecognized interaction between the experimental object and the experimental context. The experimental learning, then, was precisely the precipitation of obvious symptoms that could then be traced via diagnosis to previously unrecognized interactions between attributes of the experimental object and the experimental context.

II.4 Conclusion

In summary, I conclude that while the existing literature has acknowledged the existence of experimental trial and error in design, none of the work has tried to model experimentation or studied the dramatic impact of new experimentation techniques (e.g. computer simulation, rapid prototyping) on the economics of experimental trial and error in the design of new products and processes — a research area that will be examined by this thesis.

Finally, I would like to introduce a subtle point that will be discussed more extensively in other parts of this thesis. In the literature, there has been an increasing number of references to the importance of first-time design success in product development ("getting it right the first time") and, as a result, current managerial practice often discourages the use of iterative trial and error [with prototypes] as an effective approach to designing new products. From my early research, however, I have seen cases where designing early prototypes fast but "wrong the first time" appear to be a very effective strategy in product design. Reinertsen (1992) recognized this point in a short article where he suggested that:

> "... some companies believe the key to rapid product development is designing things right the first time, and that too much time is spent correcting mistakes. The great difficulty in designing a product perfectly the first time lies in solving new problems. Such solutions depend on constructing precise conceptual models for the problems, and these can require complex and time-consuming analysis. A more practical, traditional alternative is to solve problems through successive approximations. The fastest way to a good engineering solution is to devise simple models first and refine them if they prove inadequate. This may mean doing it wrong the first time, but doing it fast."

Thus it seems to me that there are circumstances under which first-time prototype success is very effective, but there may be other circumstances under which fast and inexpensive prototype iterations proof to be more effective. A subgoal of this thesis is to understand these circumstances and find empirical support for cases where "designing it wrong the first time" can be an effective design strategy.

Chapter III

Experimental Cycles in Design: A General Model

III.1 Introduction

In this chapter, I develop a general model of experimental cycles in design which will be used in other parts of this thesis. More specifically, the information presented in this chapter is divided in following sections:

— I start (section III.2) by modeling experimentation as an iterative four-step *design-build-test-analyze* cycle aimed at improving a design solution and discuss the activities related to each of those steps. Furthermore, I will specify the conditions internal and external to a design activity that lead to iterations in experimental cycles.

— Next (section III.3), I define the following parameters that can be used to characterize the performance of an experimental cycle: (1) the cost of conducting the four experimental steps; (2) the accuracy of the models used (i.e. the probability of identifying a quality improvement opportunity <u>if</u> it exists and is tested for); and (3) the financial quality loss (i.e. the cost of not identifying an improvement opportunity <u>if</u> it exists).

— In section III.4 (hypotheses formulation) , I propose that the efficiencies of experimentation "modes" (e.g. simulation, prototype testing) vary in two ways: (1) they vary with respect to the four steps <u>within</u> an experimental cycle; (2) as a design progresses, they decrease at different rates <u>between</u> cycles because of diminishing marginal returns. As a result, I propose that, as a design progresses, designers will find it economical to switch between modes (i.e. there exists an optimal switching point (OSP)). To understand the radical impact of new and improved modes of experimentation, I develop hypotheses related to the impact of improvements in model accuracy and model build cost on the optimal mode switching point and total design cost. Finally, I propose that experimentation involves two micro-level learning processes: (1) "interference finding" which is described by von Hippel

and Tyre (1994); and (2) learning about mode efficiencies which is fundamental to the iterative evaluation of a mode's efficiency.

III.2 A General Discussion of the Experimentation Model

Experimentation can be seen as an iterative four-step *design-build-run-analyze* cycle (see figure III.1). For example, one might (1) conceive of and design a new, more rapidly-deploying airbag for a car; (2) build a prototype of key elements of that airbag as well as any special apparatus needed to test its speed of deployment; (3) run the experiment to determine actual deployment speed; and (4) analyze the result.

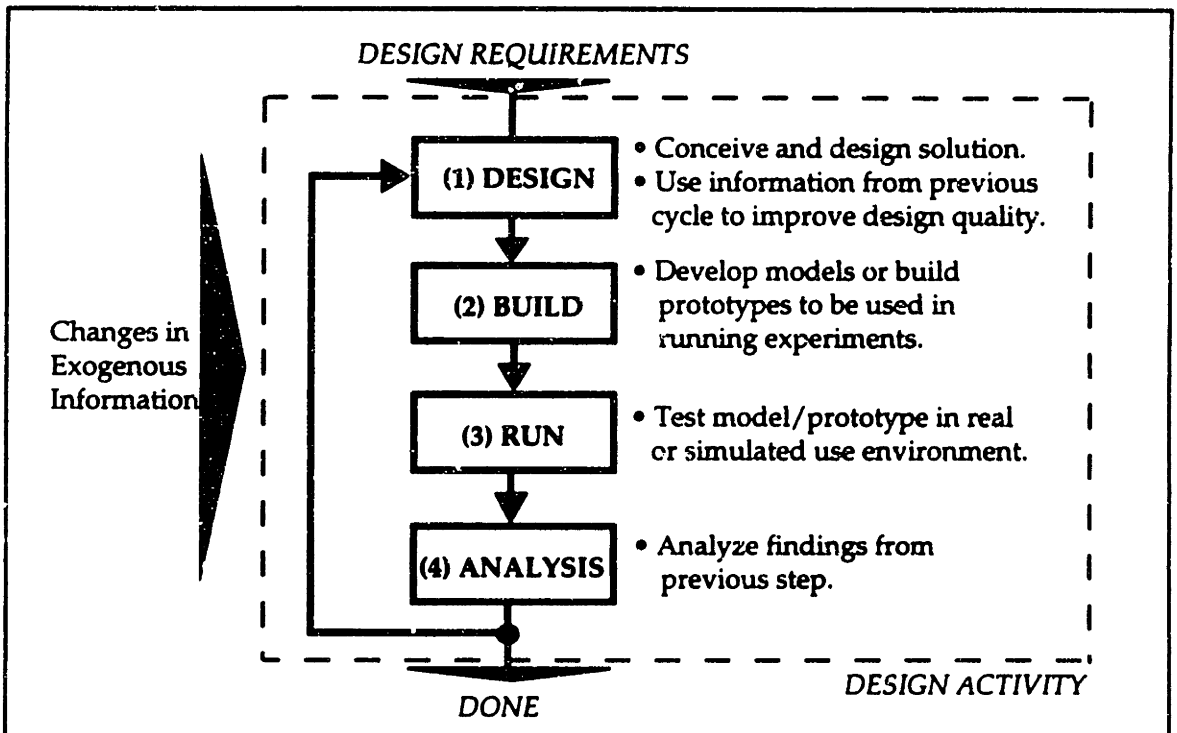


Figure III.1. The model views experimentation as an iterative four-step cycle. Changes in exogenous information are due to unstable user needs, rapid technological advances, etc.

If the results of a first iteration are satisfactory, one stops. However, experimentation is usually a matter of repeated trial and error: That is, if analysis shows that the quality of a design can be improved cost-effectively, one modifies one's design on the basis of what one has learned and "iterates" or tries again[1]. As one has the opportunity to improve the design at the end of each iteration, a complete design may involve many iterations until the incremental marginal cost to improve quality will exceed the incremental marginal benefit from the improvement. A model of this cyclical experimentation process will be part of this chapter.

In the discussion of an iterative experimentation model, I will distinguish between two types of iterations: (1) iterations caused by experimental learning that occurs inside (or endogenous to) a design activity (e.g. the designer has an imperfect understanding of a design solution being tested); (2) iterations caused by information changes that occur outside (or exogenous to) a design activity (e.g. the customer changes design specifications). In this thesis, I will look primarily at iterations that occur inside a design activity which, as we have seen in the literature review, has received little attention by past researchers.

I start the discussion in section III.2.1 under the assumption that no exogenous changes take place during the iterative convergence on a design solution. This assumption will be relaxed in the general discussion of exogenous changes (section III.2.2).

---

[1] One should also consider the possibility of degrading quality during an attempt to improve it. Sometimes, fixing an error will introduce new errors and the actual net improvement will be negative.

III.2.1 Experimentation Inside A Design Activity

In the proposed model, experimentation and the associated learning can be seen as a four-step cycle which has been depicted in figure III.1. Fundamental to experimentation is the process of design improvement which can be divided into two (interdependent) activities: (1) the detection, analysis, and removal of design errors (error elimination)[2]; and (2) the adjustment of design parameters as to improve design performance (parameter design). My empirical research will primarily focus on the role of experimentation in error elimination but I will try to keep the initial discussions general so that parameter design can be analyzed as part of future studies.

An error is a problem in the external operation of a design (operational failure) due to an internal design error. Thus, error elimination consists of finding such operational failures (detection), determine the internal design error that causes the failure (analysis), and modify the design as to remove the internal error (removal). In contrast, parameter design consists of systematic and planned changes in design parameters as to increase design performance, usually with the aid of the statistical design of experiments (Box and Draper 1989, Phadke 1989, Taguchi 1987). Error-free designs that operate without operational failure, can often be improved with parameter design. Parameter design is normally performed after design-errors are eliminated, or during later experimental cycles. (It should be noted, however, that

---

[2] Research in software design has shown that error elimination accounts for a significant part of total design cost. Boehm (1983, p.13) studied the effort distribution of several programming projects and found that about 40% of total effort is devoted to detecting errors and correcting the software to eliminate errors which are detected.

parameter design for maximizing design robustness may actually prevent errors that are a result of oversensitivity to uncontrollable noise levels. Thus a design strategy that focuses on robustness early is likely to reduce the number of iterations necessary for error elimination later.)

With design improvement as the goal of experimentation, I now discuss the content of each experimental step:

Step 1: Design

During *design*, designers are initially given a set of input and output specifications that are used to guide the design of a solution to be tested by experimentation. Often these are not optimal solutions; instead they are satisfactory[3] with respect to a given set of requirements and constraints. Once a candidate solution is developed, steps (2), (3), and (4) (build, run, and analyze) are used to evaluate that solution and identify opportunities to improve it. Only improvements found necessary as a result of analyzing a given experimental cycle will be made in the *design* phase of the next experimental cycle. Normally such improvements are of the following kind:

— *Error elimination*: During a previous experimental cycle, an error has been precipitated[4] and its cause has been found. Thus, step (1) will

---

[3] Simon (1981) noted that traditional engineering methods tend to employ more inequalities (specifications of satisfactory performance) rather than maxima and minima. These figures of merit permit comparisons between better or worse designs but they do not provide an objective method to determine best designs. Since this usually happens in the real world, Simon introduces the term "satisfice", implying that a solution satisfies performance constraints.

[4] Testing for the absence of negative qualities ("errors") rather than the presence of positive qualities as a means to economize on complexity is what Alexander (1964) describes as a mechanism to test good fit. The list of positive qualities for a design would be nearly endless and, if supplied, one would also need a description on how these qualities interact with the

involve a series of activities as to eliminate the error cause. This may involve a minor correction of an existing design solution or the complete generation of a new solution[5].

— *Parameter design*: Design performance can be further improved by adjusting the levels of key design parameters as to maximize a defined quality performance metric. The amount of adjustment is determined during a previous iteration with the help of running a set of designed experiments (step (3)), followed by a careful analysis of its results[6] (step (4)).

"field". Alexander (1964) gave the following example: "...Think, for instance, of trying to specify all the properties a button had to have in order to match another. Apart from the kinds of thing we have already mentioned, size, color, number of holes, and so on, we should also have to specify its specific gravity, its electrostatic charge, its viscosity, its rigidity, the fact that it should be round, that it should not be made out of paper, etc. In other words, we should not only have to specify the qualities which distinguish it from all other buttons, but we should also have to specify all the characteristics which actually made it a button at all. Unfortunately, the list of distinguishable characteristics we can write down for the button is infinite. It remains infinite for all practical purposes until we discover a field description of the button. Without the field description of the button, there is no way of reducing the list of required attributes to finite terms. We are therefore for[illegible] to economize when we try to specify the nature of a matching button, because we can only grasp a finite list (and a rather short one at that). Naturally, we choose to specify those characteristics which are most likely to cause trouble in the business of matching, and which are therefore most useful in our effort to distinguish among the objects we are likely to come across in our search for buttons..."

[5] Unfortunately it is very difficult to predict the difficulty of correcting errors. It is often assumed that bugs which are hard to detect are also hard to correct. In a study of 63 bugs in software design, Shooman (1983) found no evidence in support of this assumption. His data showed that there was no correlation between the difficulty of detection and correction.

[6] Parameter design uses statistical methods to analyze results. Examples range from simple significance tests (Hogg and Ledolter 1987), to response surface methodologies (Box and Draper 1989), to robust design (Phadke 1989, Taguchi 1987).

Step 2: Build

During *build,* models are built to test design solution(s) generated during step (1). Such models can have varying degrees of complexity with respect to "reality". The value of using models in experimentation is both to reduce investments in aspects of the real that are irrelevant for the experiment, and to control out some aspects of the real that would affect the experiment in order to simplify the analysis of the test results (step (4)). Thus, a model of an airplane used in wind tunnel experiments has no internal design details — these are both costly to model and [mostly] irrelevant to the outcome of wind tunnel tests. Sometimes a model to be built is incomplete because one cannot economically incorporate all relevant aspects of the "real" or does not know them. The incompleteness of a model can lead to unexpected design errors when a given model being used in testing is replaced by a different (and more accurate) model or by the real design in the real environment for the first time.

In software development, for example, a rapidly-built prototype is an early part of a system that enacts the essential features of the later system. The prototype has few internal details of the final software structure but "simulates" its use at the user interface level. Software prototypes are normally used to identify weaknesses in understanding real user requirements (Crinnion 1991; Keen 1991; Carey and Mason 1983). Wheelright and Clark (1992) note that prototypes can take many forms, ranging from

numerical simulation models (CAD, finite element, heat transfer, etc.), breadboard models, mockups, to fully-functional products[7].

Ulrich and Eppinger (1994) develop a general framework that classifies prototypes[8] along two dimensions: (1) the degree to which they are physical as opposed to analytical; and (2) the degree to which they are comprehensive as opposed to focused. Thus, the airplane prototype used in wind tunnel experiments is (1) physical ; and (2) focused since it includes only features that relate to the plane's aerodynamic behavior. Ulrich and Eppinger propose that analytical prototypes (e.g. computer simulation) are generally more flexible than physical prototypes whereas physical prototypes are required to detect unanticipated phenomena.

Step 3: Run

During *run*, the model built in step (2) undergoes a series of tests that evaluate its behavior in its intended use environment. Again, the use environment may be a model representing the "real" environment which was built during step (2). Thus in the example of automobile crash

---

[7] Wall, Ulrich, and Flowers (1991) have developed a framework that allows product development professionals to evaluate different prototype technologies. The measures they propose are part performance (an aggregate measure of the fidelity of the properties of a prototype part, made with a particular process, with respect to the properties the part would have if it were made with the intended final production process), unit cost, and lead time. They use the framework to evaluate four different prototyping technologies (computer-aided design solid modeling, stereolithography, computer-numerically controlled machining, and room vulcanizing rubber molding) for various development processes at Eastman Kodak. Wheelright and Clark (1992) and Clark and Fujimoto (1991) use similar criteria to evaluate cost and representativeness of different forms of prototypes.

[8] They define prototypes as an approximation of the product on one or more dimensions of interest.

simulations, a mathematical model of a car may be crashed into the mathematical model of a brick wall.

Test strategies can involve any or a mix of the following:

— *Error precipitation:* A test is run until an error is precipitated, or is stopped if no errors are found after a period of testing. (The criteria for establishing when a test is complete varies. An example of such a criteria is the period that corresponds to the expected "lifetime" of the product being designed.) The likelihood of detecting an error is a function of two variables: (1) the accuracy of a given model with respect to the "real" (or final) product; and (2) the proportion of test cases relative to the total number of cases anticipated ("test coverage"). Thus an increase in the likelihood of detecting design errors can be achieved by (1) investing in more accurate models; and (2) increasing test coverage. But both of these approaches may have significant limitations. Consider the use of computer simulation and prototype testing in the detection of errors. Computer simulations can rarely be exhaustive because of the significant discrepancy between computer speed and real-time execution of system behavior. (In integrated circuits, the discrepancy is several orders of magnitude, even for the most powerful computers.) Thus, if a high degree of test coverage is needed, it may be economical to switch to a physical prototype that can run tests in real-time instead of trying to increase test coverage by additional investments in simulation.

— *Parameter design:* Once a set of experiments is designed (step (1)) and the test set-up is built (step (2)), a series of experiments at different

parameter settings is run and the effect on selected performance metrics measured. The number of experimental runs is often small as modern experimental design techniques have been developed to economize on experimental trials by compromising the data collected for analysis. For example, resolution III designs (also known as orthogonal designs) require the fewest number of experimental runs since interaction effects between parameter settings are confounded with their main effects. Under such circumstances, a statistical analysis of interaction effects in step (4) cannot be performed.

Step 4: Analysis

During *analysis*, designers analyze the information that was acquired from running tests in step (3) so as to identify opportunities for improving a candidate design solution. Again, the activities can be divided into two categories:

— *Error elimination*: Since an error was detected during the previous step, the designer follows a series of diagnostic steps in order to identify the error cause(s). Sometimes the designer has a thorough understanding of the tested prototype and finds the error cause very quickly. Very often, though, subtle errors make the analysis very difficult, especially in cases of great complexity and poor knowledge of causal relationships between system inputs and outputs. As a result, designers tend to rely on diagnostic tools and problem-solving methods to aid in their analysis of error symptoms. A very effective analysis tool is the use of computer simulation since it gives a designer quick access to virtually

any type of information within the realm of the underlying model. In contrast, an analysis of data from prototype testing is more difficult since access to error-related information is typically limited. For example, consider that a real car crash happens very quickly — so quickly that the designer's ability to observe details is typically impaired, even given high-speed cameras and well-instrumented cars and crash dummies. In contrast, one can instruct a computer to enact a virtual car crash as slowly as one likes, and can zoom in on any structural element of the car that is of interest and observe the forces acting on it and its response to those forces during the crash.

—*Parameter design*: In parameter design, the analysis of test results is often straight-forward because designer know what type of information is needed prior to starting the analysis. The test data from step (3) (run) is typically analyzed with standard statistical models such as the analysis of variance (ANOVA). Experimental runs in step (3) are usually designed as to maximize the efficiency of analyzing its results in step (4) while minimizing the number of experiments to be conducted. (For example, the use of symmetry in designing an experimental matrix greatly enhances the efficiency of a later analysis.)

III.2.2 Iterations Due to External Changes

The presented four-step model is iteratively converging on a design solution. However, one sometimes finds that this convergence process can be affected by changes in the information exogenous to a design activity. Such changes can have a significant impact on innovation cost and time as

innovators may find it necessary to modify designs that are near completion. Or in some cases, innovators may be forced to "start over" as a sudden change renders their design solution unsuitable. In both cases, changes will trigger an additional sequence of experimental cycles in order to accommodate new or modified requirements[9]. Thus the incremental cost of responding to exogenous changes is affected by the incremental cost to conduct an experimental cycle (which was addressed in the previous section). In my research of innovation processes, I have found that most observed changes can be attributed to the following causes[10]: (1) rapid technological changes during the development process renders a selected solution obsolete; (2) as

---

[9] As described during the literature review in chapter II, Krishnan and Eppinger (1993) have developed a model-based framework of overlapping coupled product development activities. In their model, iterations occur due to the downstream activity's early absorption of incomplete upstream information.

[10] The problem of unstable requirements was described in our field research on hardware design. I found the following two examples as part of my initial field research:

(a) Changes in technology:

Personal computer circuit boards normally consist of an array of integrated circuits (ICs). The central processing unit (or CPU) is the "brain" of a computer and the remaining ICs usually execute support functions, such as controlling external and internal devices. A design engineer's group had the task of integrating a large number of these supporting ICs onto a single chip. A critical feature in the design was the CPU's clock frequency because it determined the speed requirements of the design. Well into the design, the manufacturer of CPUs (Intel) introduced a revised microprocessor that allowed personal computers to run faster. The CPU's clock frequency increased by 50% which had a serious impact on the design and integration of supporting ICs. The change forced the design group to modify their existing design which eventually added 15% to their total development time.

(b) Changes in user needs:

In another interview, a designer of a video board reported a case where customers changed their needs after using the product. Before the design project was started, customers were interviewed about their color preferences in a video application. Customers indicated that they found color X to be the most appropriate selection. However, after the first prototype was available and customers actually saw the color on a screen, they felt that their selection was poor. Instead they changed the requirement to color Y which resulted in a design change of the video board.

For more case evidence, see also Jones (1992) and Tallyn (1990).

users are engaged in problem-solving themselves, their needs are unstable; and (3) as designers are engaged in problem-solving, they often change solution paths as a design evolves. (And, since designs are often partitioned into multiple interdependent design activities, changes in one design activity can cause iterations in other activities (Smith and Eppinger 1991, 1992).)

Studies of the engineering problem-solving process have shown the process of creating and eliminating needs for components during the process of system development quite clearly. Marples (1961) explored the problem-solving process involved in some industrial process equipment development projects. Allen (1966) studied the problem-solving process involved in the design of a number of aerospace systems. Both found that engineers carried out the work of developing a system or subsystem by conceiving of and evaluating a number of alternative designs, and then selecting the one judged best. Alternative designs involved different components: When the system to be built was finally settled upon, the components in the selected alternative were "needed", while those embodied in the rejected alternative were not. Allen also showed that engineers' preferences for alternative solutions (and therefore preferences for related component products) changed frequently and quickly as development work proceeded. He displayed these attributes of the problem-solving process graphically in "solution development records" (figure III.2 on the next page).

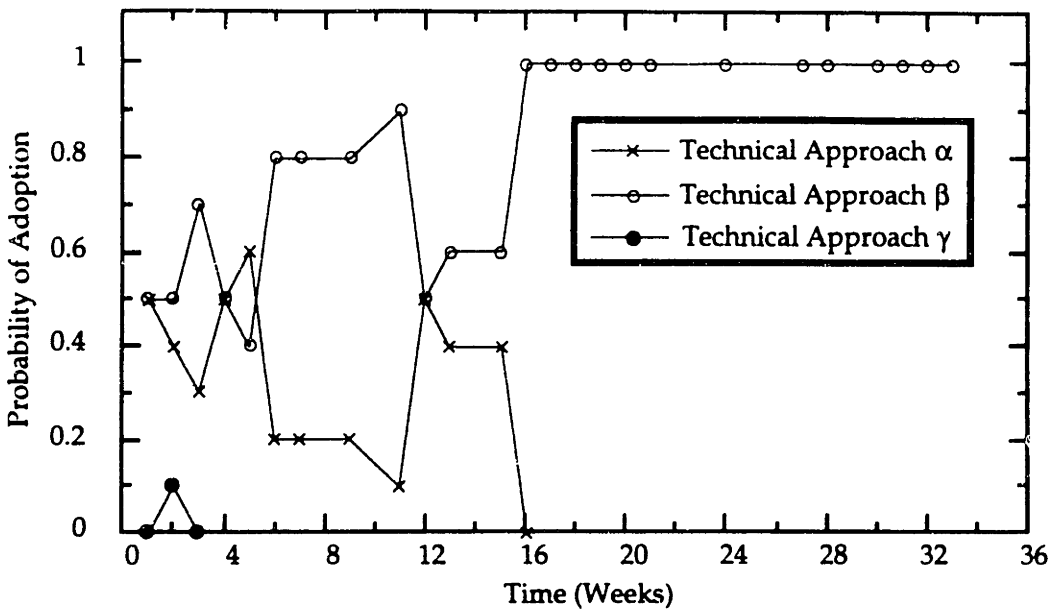


Figure III.2. A "Solution Development Record" of an engineering design team designing an Antenna Radiation Subsystem shows a rapid change over time in the probability that one of the three technical approaches (α,β, and γ) will be adopted. These records are based on data from real-time monitoring of the engineering problem-solving process. (Redrawn from Allen 1966.)

III.3 Description of General Parameters

In this section, I will define parameters that influence the efficiency of experimentation and, as a consequence, the designers' decisions in selecting experimentation modes.

III.3.1 Cost

An experimental cycle consists of four steps (design, build, run, and analyze) and each step has a cost associated with it. (The term 'cost' refers to monetary cost and/or the cost of time.) Let $\vec{C}_i$ represent the cost vector at iteration i, where:

$$\vec{C}_i = \begin{bmatrix} C_i^{design} \\ C_i^{build} \\ C_i^{run} \\ C_i^{analyze} \end{bmatrix} \qquad \text{(Equ. III.1)}$$

The cost of each step is now discussed in turn.

— *Design*: The *design* step consists of using information from a previous iteration in order to make changes (and verify these changes). The cost of making such changes can depend on a number of variables: the underlying technology, the structure of the design, the designer's skill, the availability of good design tools, etc. For example, designs that consist of highly independent subsystems are easier to modify than designs with interdependent subsystems (Alexander 1964, von Hippel 1990).

— *Build*: The cost of building a model (computer simulation or prototype) depends highly on available experimentation technology and the degree of accuracy that the model is intended to have[11]. For example, modern computer-aided design (CAD) tools can often interface to computer software that converts a design directly into a simulation model. In such cases, the cost of building a model consists mainly of the investment in conversion tools (fixed cost) and the effort required to operate them (variable cost). Similarly, in cases where physical

[11] The tradeoff between the quality and the cost of an iteration is discussed by Chao (1993). In two case studies, she investigates design iterations and the time/quality tradeoff in the VLSI (Very-Large Scale Integration) circuit product life cycle. She proposes two strategies to manage the time/quality tradeoff: (1) faster iterations; and (2) higher quality iterations. Using quantitative data from the case studies, she demonstrates these two strategies.

prototypes are required (i.e. the use of computer models is ineffective), long prototyping lead times have been dramatically reduced by the emergence of rapid prototyping technologies.

— *Run*: The cost of running a series of tests will depend on the speed at which the tests can be run and the number tests to be run.

— *Analysis*: The cost of analyzing the results from step (3) depends to a significant degree on access to test-related information and the availability of tools that aid in the problem-solving process. In parameter design, the type of information required for analysis is known prior to the experiment and therefore the cost of analysis tends to be low and predictable. In error elimination, however, designers often do not know what kind of information is necessary to find the error cause. As a result, having quick access to any information will significantly affect the total time required to complete an analysis.

III.3.2 Accuracy

The second variable is the degree of accuracy of a given experimentation mode with respect to the "real" (and final) product and its intended use environment. I define accuracy as the probability of identifying an improvement opportunity in a given experimentation mode if that opportunity is tested for (i.e. it excludes the effect of limited test coverage).

More formally, I define:

Let: $\vec{\alpha}_i$ = accuracy of model at iteration i (Equ. III.2)

Accuracy[12] is an important factor during all experimental steps but particularly affects step (3) (run) and step (4) (analysis). For example, a simulation model that is an inaccurate representation of the "real" (or final) system will not be able to detect some design errors, irrespective of the number of tests conducted. Such errors will only be detected by either improving the accuracy of a simulation model in-use (which would increase the cost of step (2)) or by switching to a different experimentation mode with higher accuracy (which would affect the efficiency of conducting other steps).

III.3.3 Quality Loss

Experimental cycles are repeated to improve design quality. The ideal design quality a customer can expect is that every product delivers the target performance each time the product is used, under all intended operating conditions, and throughout its intended life, with no harmful side effects (Phadke 1989). Thus a designer will consider additional experimental cycles if the experiments are expected to reveal new information that can be used to improve the quality of a candidate design solution. In contrast, not conducting additional experimental cycles will result in a financial quality loss[13]. Similarly, conducting additional cycles but failing to identify existing opportunities to improve quality results in a financial quality loss. As a

---

[12] Please note that we defined accuracy as a vector. Since designers may be interested in many dimensions of a design, models are sometimes developed to be accurate only with respect to subset of all dimensions known or considered.

[13] The term "quality loss" is borrowed from Taguchi's work on quality engineering (see Phadke 1989).

consequence, the decision to conduct additional experimental cycles depends partially on a designer's expectations about the financial quality loss.

More formally, I define:

Let: q_i = expected financial quality loss at iteration i (Equ. III.3)

In the case of using computer simulation to eliminate design errors, the quality loss of not detecting an error is related to the financial loss incurred if this error causes an operational failure in a prototype or in the field. If the expected quality loss in a prototype or the field is high, designers may find it economical to perform additional experimental cycles with computer simulation to identify the error before proceeding to prototype testing or a field release.

III.4 Hypotheses Formulation

With the four-step model (design, build, run, analyze) and its performance parameters (quality loss, cost, and accuracy) defined, one can now begin to develop hypotheses to be tested in an empirical setting.

III.4.1 Variations in Experimentation Efficiencies and Mode Switching

When one compares the efficiency of different experimentation modes, one will find two types of variation: (1) variation within an experimental cycle; and (2) variation between experimental cycles. I will discuss both in turn.

Within-Cycle Variation

In the previous section, we have seen that the efficiency of experimentation modes differs with respect to the four steps within an experimental cycle (*within-cycle variation*). For example, consider the example of using simulation and prototypes in the crash testing of automobiles. As explained earlier, simulation can be very efficient with respect to the analysis of crash data (step (4)) and the building of a model (step (2)) but can be inefficient (i.e. slow) and inaccurate with respect to running the actual crash test (step (3)). In contrast, automobile prototypes are very costly to build (step (2)) and their crash data can be difficult to analyze (step (4)) but they tend to be more accurate and can be crashed very quickly (step (3)).

Thus, the first hypothesis follows:

Hypothesis 1: Experimentation modes (e.g. simulation, prototype testing) vary with respect to the efficiency of conducting the four experimental steps within an experimental cycle.

Between-Cycle Variation

As a design progresses and experimental cycles are repeated, I propose that the efficiency of experimentation decreases due to diminishing marginal returns from experimenting in a selected mode, and that different modes decrease at different rates. In other words, an experimental mode's efficiency may be very different for iterations early and later in a design. This dynamic variation in efficiency will be referred to as the *between-cycle* variation.

For example, consider the use of computer simulation in the detection of errors in integrated circuit design. As the cumulative number of design errors eliminated is increasing, the pool of residual errors tends to decrease. Therefore the mean time between errors detected increases which implies that the number of tests to be run until another error is detected also increases. The overall effect is a monotonically decreasing error detection rate, resulting in an increase in run time cost for a given return[14]. (For a simple proof of this relationship, see appendix A.) Thus, the marginal return (errors detected per unit effort) from simulation diminishes as a function of experimental cycles. Similarly, prototype testing will experience the same type of diminishing returns but, because prototypes run tests significantly faster than simulation, at a slower rate (see figure III.3).

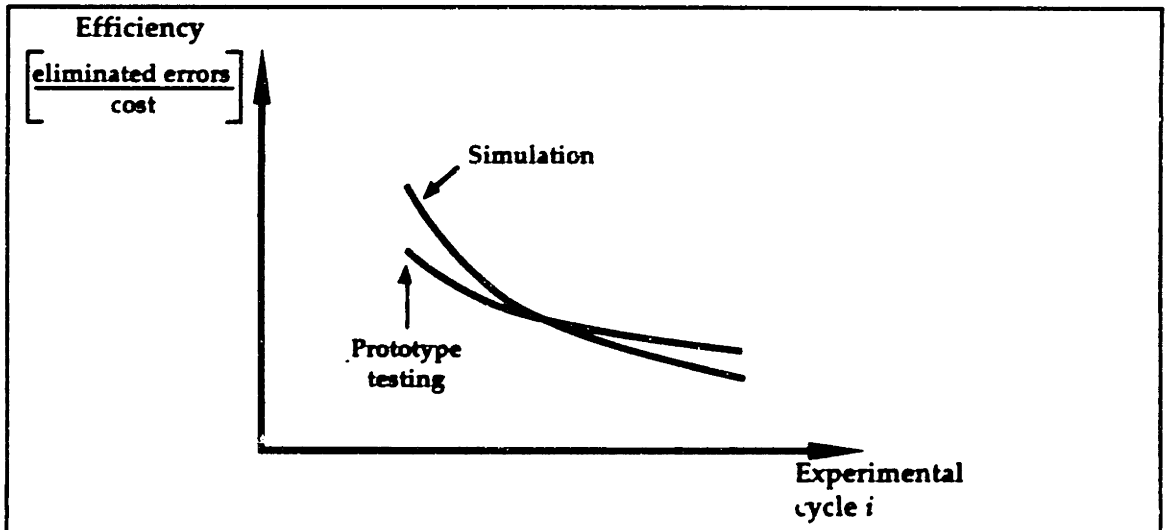


Figure III.3. The decreasing efficiencies of simulation and prototype testing (illustration).

[14] There is considerable evidence on the described relationship (cumulative errors and debugging time) in the literature on software design (Boehm 1981, Shooman 1983).

Thus, the second hypothesis follows:

Hypothesis 2: The efficiencies of repeated experimental cycles in selected modes (e.g. simulation, prototype testing) decrease at different rates.

Mode Switching as an Experimentation Strategy

As the efficiency of experimentation modes varies within and between experimental cycles, one would conceivably start experimentation with the mode that is most efficient under initial experimental conditions. Then, as a design progresses and the mode's marginal returns diminish, there may be a point ("the switching point") where switching to another [more efficient] experimentation mode will be an economic strategy. Such an experimentation strategy will be referred to as "mode switching".

As a general example, consider the use of simulation and prototype testing in integrated circuit design once more. Earlier in this chapter, I have pointed out that computer simulation is usually more efficient with respect to experimental step (2) (building a test model), step (4) (analysis of error symptoms), and step (1) (error removal). In contrast, prototypes are more accurate and can execute test cases (step (3)) much more rapidly than simulation can. But since the number of design errors is usually high prior to starting experimentation, the mean-time between errors detected tends to be small and thus most designers will find it economical to start with simulation. As a design progresses and the error detection rate of simulation declines, the incremental efficiency of continuing to simulate will at some point fall below the incremental efficiency of prototype testing. Thus,

designers will find it economical to switch from simulation to prototype testing where their efficiency trajectories intersect (the optimal switching point (OSP)), and continue iterative experimentation with prototype testing (illustrated in figure III.4). I submit that finding the optimal switching point can result in significant improvements in experimentation cost.

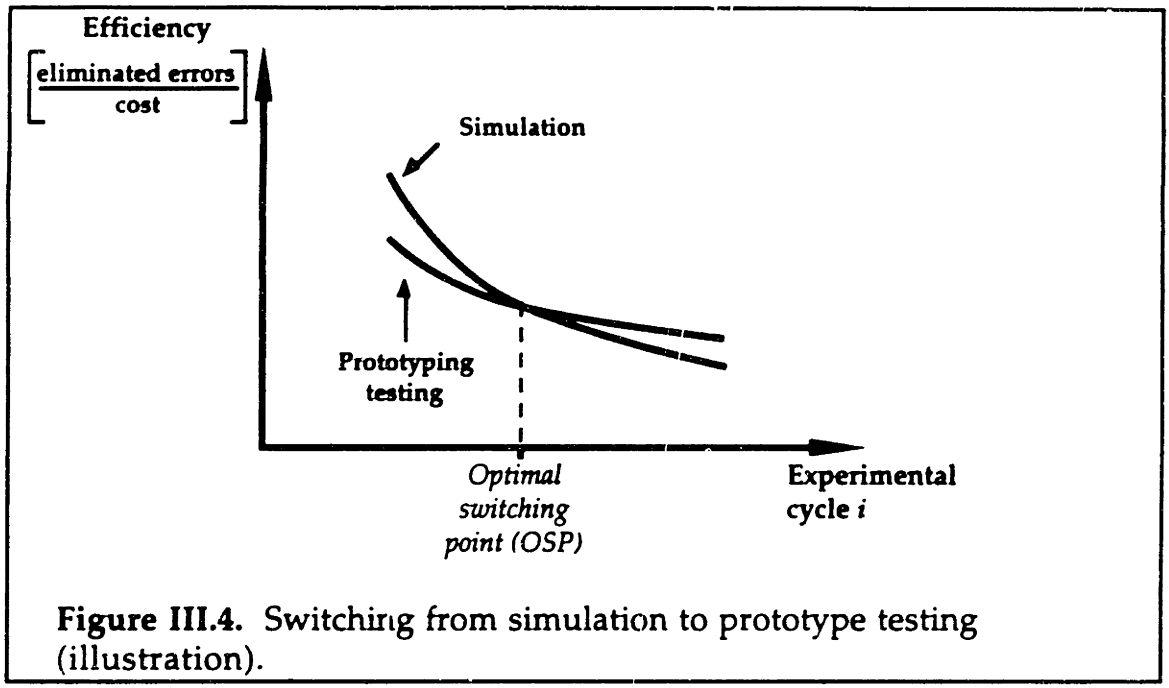


Figure III.4. Switching from simulation to prototype testing (illustration).

Proposition 1: As a design progresses, designers will find it economical to determine the optimal switching point (OSP) between experimentation modes and revise their switching strategies accordingly.

Unfortunately, design practitioners often do not take advantage of or actively manage the process of experimental mode switching. Some of the deficiencies that I observed in my research on circuit design are: (1) designers do not have decision support tools that aid in the evaluation of mode efficiencies but have to rely mostly on intuitive and tacit rules they have

developed from experience; and (2) organizational routines and procedures sometimes discourage an objective evaluation of the optimal mode switching point (e.g. designers are rewarded for first-time prototype success although it may be suboptimal to follow such a design strategy) . I will address deficiency (1) by developing and applying a decision model to mode switching in chapter IV. Deficiency (2) will be addressed under implications for managerial practice in chapter VII ("Summary and Conclusions").

III.4.2 The Impact of Exogenous Innovations on Mode Switching and Design Performance

In the thesis introduction (chapter I) I proposed that the economics of experimentation is being radically affected by the use of new and greatly improved versions of methods such as computer simulation, mass screening, and rapid prototyping. I will now hypothesize on how some of these radical changes have affected optimal mode switching strategies.

The Impact of Variations in Model Accuracy on Mode Switching

When one considers the impact of variations in accuracy on experimental steps, one finds that the steps most affected are *run* (step (3)) and *analysis* (step(4)). Please recall that during *run*, a designer runs experiments in order to identify opportunities for improving the design quality. In the case of running tests to detect design errors, there are two reasons why an error may go undetected: (1) the experimental model is inaccurate with respect to the tested behavior; and (2) not enough test cases are run (incomplete test coverage). Thus, changes in accuracy will have an impact on

reason (1) and the efficiency (errors detected per unit cost) of conducting step (3) in the experimental cycle. Similarly, errors that fall outside the modeled design behavior cannot be analyzed which affects (step (4)). As a result, changes in accuracy will have an impact on a mode's efficiency with respect to a complete experimental cycle. Assuming that these changes in accuracy come from innovation sources that are exogenous to a particular design process[15] (*ceteris paribus* conditions), I propose that they result in a shift of the optimal switching point. Figure III.5 illustrates a shift in the OSP between simulation and prototype testing, given that an exogenous innovation has resulted in an improved accuracy of simulation.

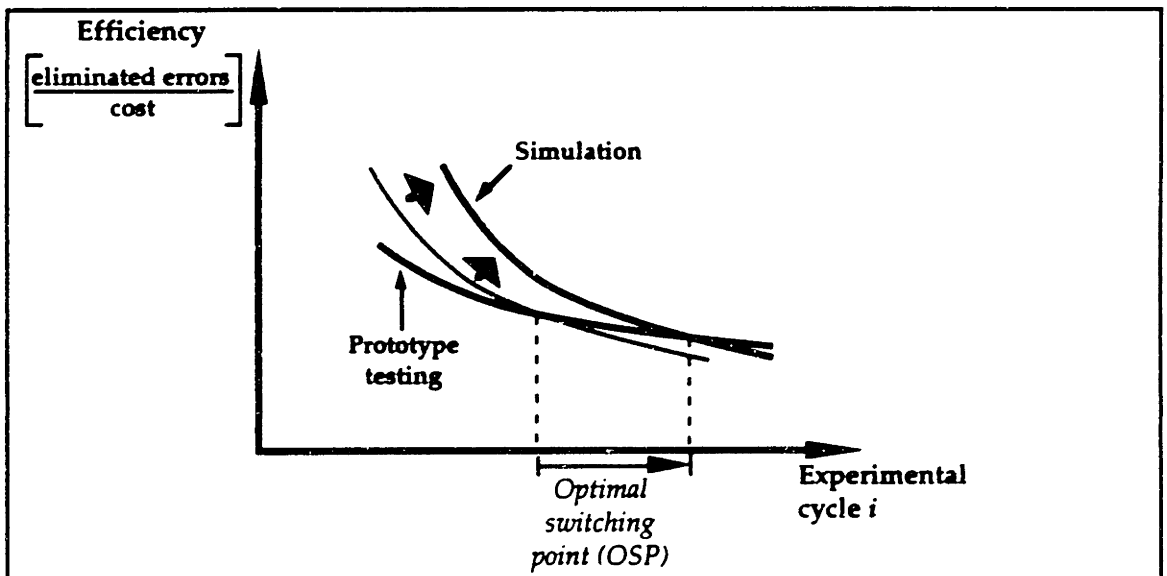


Figure III.5. The impact of increases in the accuracy of simulation models on the switching point (illustration).

As a general example, consider the dramatic changes that computer simulation has experienced in recent years. With the transition from slow

[15] Changes in accuracy due to a higher investment in step (2) (build) are excluded here.

first generation personal computers to high performance workstations (at constant cost), simulation models could be greatly improved in accuracy[16]. At the same time, we have observed an increased use of computer simulation (relative to prototype testing) in many areas, ranging from airplane design to basic science. Using the reasoning presented in figure III.5, I submit that this shift in the switching point away from prototype testing has been partially caused by the [exogenous] changes in the accuracy of simulation models.

More specifically, I propose the following hypothesis:

Hypothesis 3: Changes in the model accuracy (*ceteris paribus*) causes a movement of the experimental mode's efficiency trajectory and, as a result, a shift in the optimal mode switching point (OSP). Thus, in the case of switching between simulation and prototype testing, accuracy improvements in the former will shift the OSP away from the latter.

The Impact of Variations in Build Cost on Mode Switching

When one considers the impact of variations in the cost of building (or modifying) a test model, one finds that the step most affected is *build* (step (2)). Please recall that during step (2), models are built to test design solutions generated during step (1) (design). Thus one would expect variations in the cost of building the test model to have an effect on the cost of conducting an experimental cycle, and ultimately the efficiency of the cycle. (Again, I assume that these changes are due innovation sources exogenous to a

[16] For example, the accuracy of finite element methods can be increased by using a larger number of elements which, in turn, increases the total computation time. Thus, increases in computing performance can be utilized by (1) running faster simulations with the same accuracy; or (2) running more accurate simulations with the same computation time.

particular design process and that changes in model accuracy are negligible.) As a result, I propose that changes in the cost to build (or modify) a test model will result in a shift of the optimal switching point (OSP). Figure III.6 illustrates a shift in the OSP between simulation and prototype testing, given that an exogenous innovation has resulted in a reduced *build* cost for prototypes.

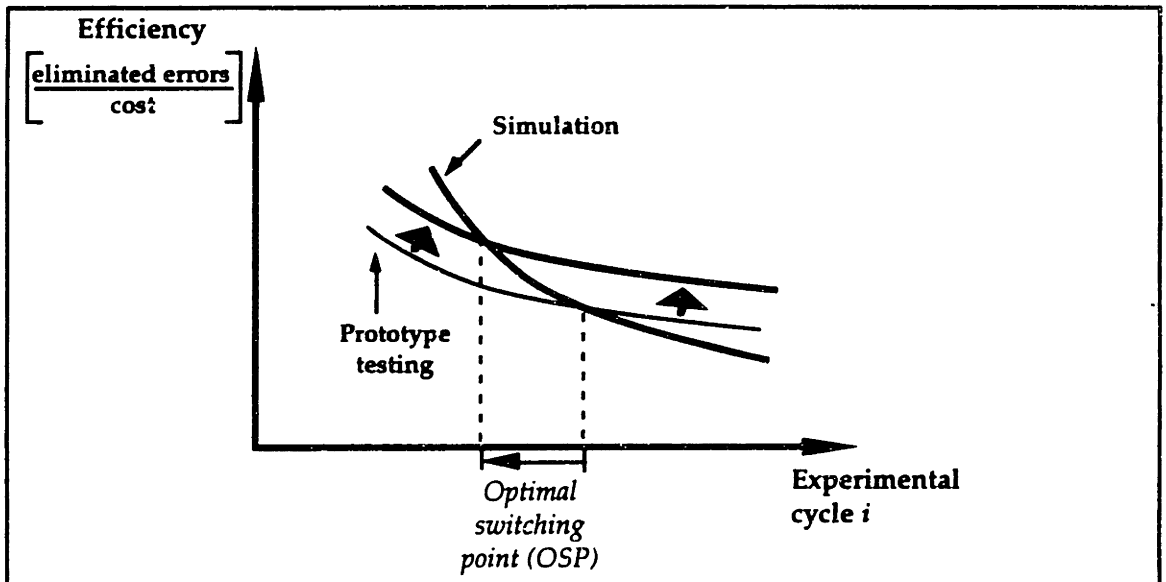


Figure III.6. The impact of a decreases in the cost of building prototypes on the optimal switching point (illustration).

As a general example, consider the dramatic improvements in the cost and speed of prototyping that have resulted from "rapid prototyping" technologies. By significantly reducing the cost and time to build a physical model prototype of a design, it has significantly reduced the cost of step (2) (build) of an experimental cycle. At the same time, we have observed an increased use of rapid prototyping (relative to other experimentation modes) at earlier points in a design process. Using the reasoning presented in figure

III.6, I propose that this shift in the OSP in favor of rapid prototyping has been partially caused by the [exogenous] change in the cost to build prototypes.

More specifically, the following hypothesis is proposed:

Hypothesis 4: Changes in the cost to build (and/or modify) a model (*ceteris paribus*) causes a movement of the experimental mode's efficiency trajectory and, as a result, a shift in the optimal mode switching point (OSP). Thus, in the case of switching between simulation and prototype testing, cost reductions in the latter will shift the OSP away from the former.

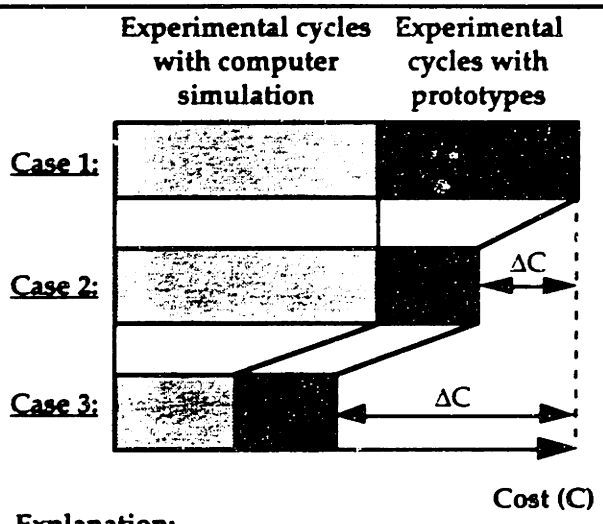
The Impact of Exogenous Innovations on Design Cost

In hypotheses 3 and 4, I have proposed that the optimal mode switching point can be shifted by exogenous innovations that lead to changes in a mode's experimental accuracy and/or cost. I will now go one step further and suggest that designers who adjust their mode switching strategies in the direction of the new optimal switching point will benefit by significantly reducing their total design cost. In contrast, designer who do not adjust their mode switching strategy may still benefit from an innovation but to a much lesser degree.

Thus, the following hypothesis is proposed:

Proposition 2: As exogenous changes in model accuracy and/or model building cost lead to a shift in the optimal switching point (OSP), designers will find that adjusting their mode switching strategies in the direction of the new OSP will result it significant reductions of total design cost.

As a general example, consider the effect of rapid prototyping on a hypothetical design process XYZ that consists of two phases: (1) experimental cycles with computer simulation; and (2) experimental cycles with prototypes (see case 1 in figure III.7). Since rapid prototyping allows a designer to build (and modify) a prototype at a lower cost, the use of rapid prototyping would most certainly reduce the total experimentation cost required for prototype testing, even if the mode switching strategy remained unchanged (see case 2 in figure III.7).



Explanation:

Case 1: Before low cost prototyping technology is available.

Case 2: After technology is available but no modification of the mode switching strategy.

Case 3: After technology is available but with modification of the mode switching strategy.

Figure III.7. The impact of rapid prototyping on design cost (with and without modifications in the mode switching strategy) (illustration).

In hypothesis 4, I have proposed that the use of rapid prototyping would result in a shift of the OSP in favor of prototype testing but away from computer simulation. Thus I propose that a designer can achieve more significant performance improvements of XYZ by reducing the amount of simulation and switching to prototype testing earlier (i.e. adjusting the experimentation strategy towards the new OSP) (see step 3 in figure III.7).

III.4.3 Learning about Mode Efficiencies

In the literature review (chapter II), we have seen that researchers have started to investigate the micro-level mechanisms of learning by doing. Thus, von Hippel and Tyre (1994) identified "interference finding" to be a form of learning by that appears to be central to the discovery of 27 problems that affected two novel process machines in their first years of use in production.

In my proposed model of repeated experimentation, I can identify two micro-level process of learning that require doing: (1) error discovery via "interference finding" as described by von Hippel and Tyre (1994); and (2) learning about mode efficiencies and their changes with respect to repeated experimental cycles. Due to its novelty, I will focus my empirical study in this thesis on (2).

As a general example of how designers learn about mode efficiencies, consider the use of simulation and prototype testing in integrated circuit design once more. Prior to starting simulation, designers have some knowledge about the initial efficiency of simulation relative to prototype testing since they have designed and tested (sometimes similar) integrated

circuits before. Thus, their experience may tell them that error analysis matters a great deal at the beginning of repeated experimentation — and simulation is very efficient in the analysis of design errors whereas prototypes are not. Their experience may also tell them that the efficiency of simulation decreases with repeated experimentation as it becomes increasingly difficult to detect errors. However, since they do not know the design errors to be detected and the resulting design changes in advance, they are unlikely to know the exact efficiency trajectories of simulation and prototype testing. As the design progresses, however, and errors are eliminated during experimental cycles, designers continuously learn about efficiency changes along with "interference finding". (For example, they may try to track the changes in the error detection rate of simulation (see figure III.8). As they expect the rate to decrease, they will find such a measure to be useful in the iterative evaluation of the efficiency of simulation.)

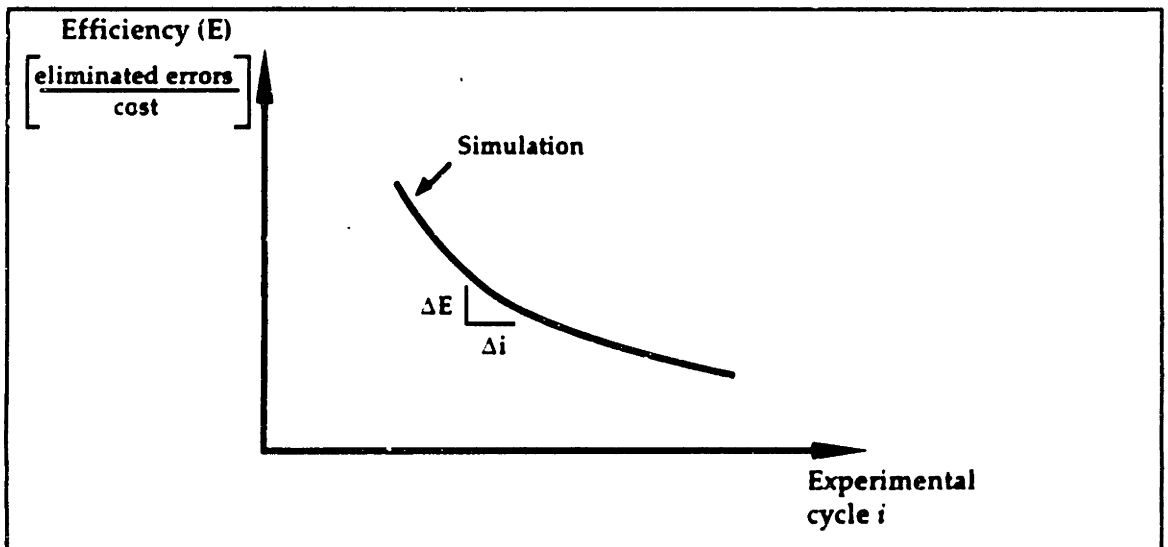


Figure III.8. Learning about the efficiency of simulation (illustration).

Thus, I propose the following hypothesis:

Hypothesis 5: Prior to starting experimentation, designers have very limited knowledge of the modes' efficiency trajectories. As the design progresses, they will iteratively "learn" about mode efficiencies from the experiments conducted.

III.5 Conclusion

An iterative experimentation model with four steps (design, build, run, and analyze) and three performance parameters (cost, accuracy, and quality loss) has been developed and discussed in this chapter. An analysis of the model has resulted in the formulation of five hypotheses and two propositions which will be refined and tested in the empirical parts of this thesis (chapters V and VI) (the preliminary hypotheses structure is shown in figure III.9 on the next page). A complete decision model that analyzes the cost and benefits of mode switching and is based on the experimentation model presented here, will be developed as part of chapter IV.

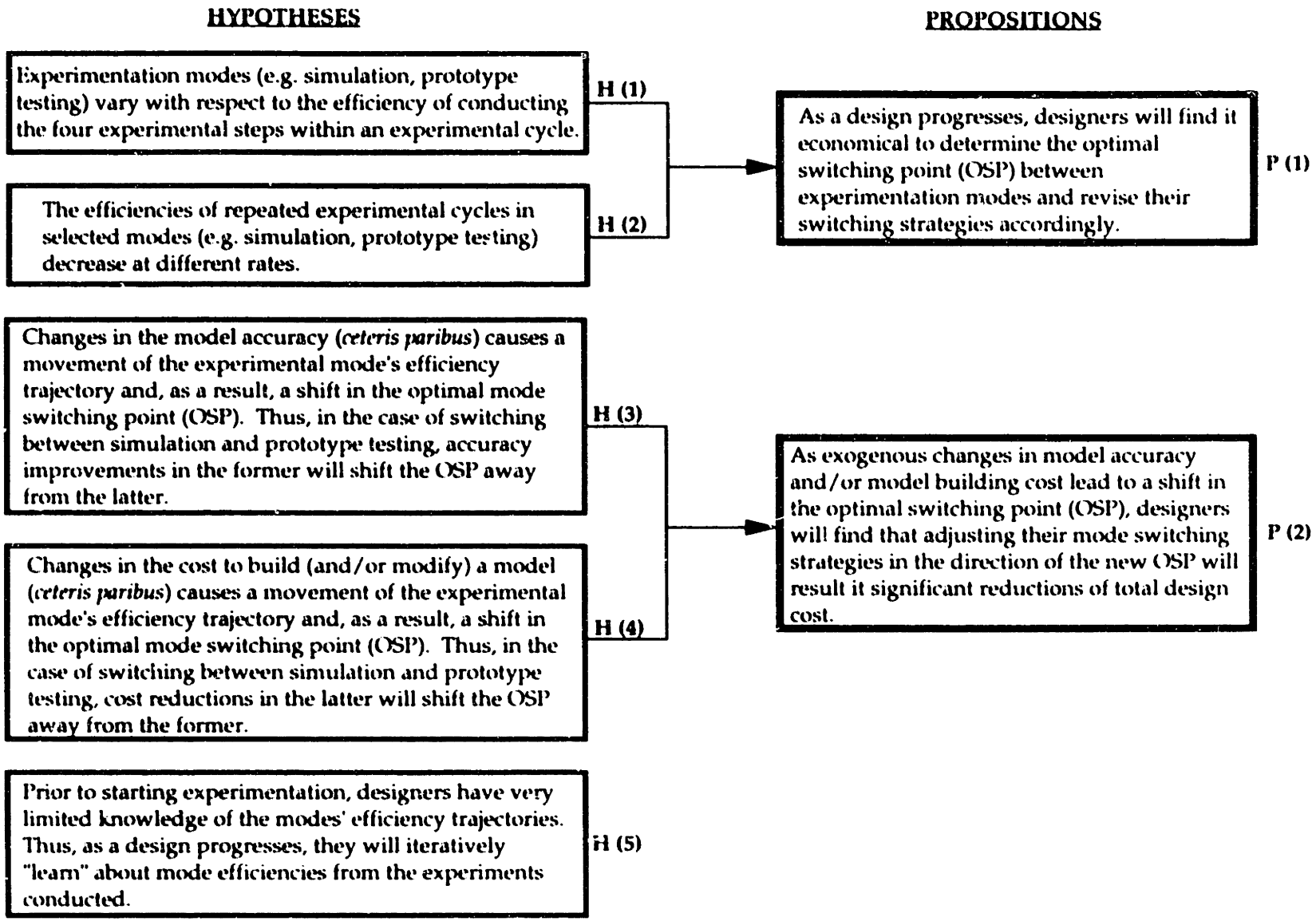


Figure III.9. Hypotheses structure to be tested.

Chapter IV

Experimental Mode Switching in Design: A Decision Model

IV.1 Introduction

In the previous chapter, I have proposed that the efficiencies of repeated experimental cycles in selected modes (e.g. simulation, prototyping) decrease at different rates. Thus, as a design progresses, designers will find it economical to determine if and where their efficiency trajectories intersect (the optimal switching point (OSP)) and revise their switching strate ; accordingly.

In this chapter, I will develop a general decision model that aids in the dynamic evaluation of a mode's efficiency and helps in understanding the process of switching between experimentation modes. More specifically, I divided the chapter in the following sections:

— I start (section IV.2) by providing some general background on decision-modeling, with a particular emphasis on the integration of bounded rationality (since one finds that designers make decisions that are boundedly rational) and iterative [Bayesian] learning .

— Next (section IV.3) I integrate the general decision model and the experimentation model developed in chapter III. The result, combined with my general understanding of design, is a decision model that gives a designer a systematic approach for [repeatedly] evaluating the efficiency of experimentation modes.

I propose that this model is more efficient with respect to how designers actually decide when to switch, but at the same time simple enough to be adopted as a useful decision support tool. Specific applications and refinements of the model will be included as part of my field research in chapter V.

IV.2 Developing A General Behavioral Decision Model

The purpose of this section is to brief the reader on decision modeling since the model derived later is directly taken from this general concept. The reader who already has a good grasp of elementary decision theory can skip directly to section IV.3.

IV.2.1 Terms and Definitions

In constructing a general decision model of rational behavior, the following terms will be used:

— *Choices of alternative behavior:* Decision-makers have a set of behavioral alternatives that they can choose from (i.e. the decision variable). These alternatives will be represented as the point set A.

— *Considered alternatives*: The subset of alternatives that a decision-maker considers when evaluating the available choices. In other words, decision-makers may have cognitive limitations or partial access to choice-related information which results in the consideration of a subset of A. I will refer to the subset as $\bar{\bar{A}}$ (where $\bar{\bar{A}} \subset A$).

— *Possible outcomes*: The set of possible outcomes if a behavioral alternative a is to be chosen. The list of discrete outcomes is a point set denoted by Ω.

— *Considered outcomes*: Similar to $\bar{\bar{A}}$, one can reasonable assume that only a subset of possible outcomes may be considered by the decision-maker. This subset is denoted by $\bar{\bar{\Omega}}$(where $\bar{\bar{\Omega}} \subset \Omega$).

— *Expected pay-off function:* The expected value to the decision-maker for every possible outcome under a considered behavioral alternative a. This pay-off function is denoted by $U(\omega)$ and is defined $\forall \omega \in \Omega$. In general, I will refer to $U(\omega)$ as the cardinal (or ordinal if only relative preferences are known) utility for each outcome ω.

— *Probability of outcomes:* The probability that an outcome ω occurs given that an alternative a is chosen. In a discrete event state, this probability is denoted as $p_a(\omega)$ (where $\sum_{\omega} p_a(\omega) = 1$ and $0 \le p_a(\omega) \le 1$).

An example of a model with two behavioral choices and three outcomes per choice is shown in figure IV.1.

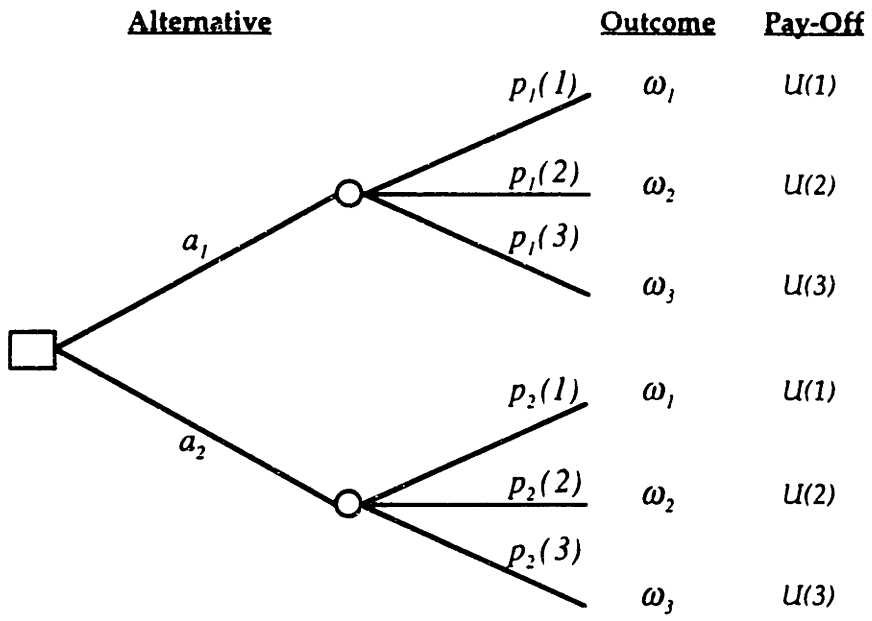


Figure IV.1. Example of a behavioral decision model for two choices and three possible outcomes per choice.

I would like to draw special attention to the point sets $\bar{\bar{A}}$ and $\bar{\bar{\Omega}}$. Traditional models view a decision-maker as perfectly rational being with unlimited cognitive abilities and access to global information. This view has been successfully challenged by Simon (1983). In addition to observing that decision-makers only consider subsets of all possible alternatives and outcomes, Simon suggested that a decision-makers ability to assess pay-offs and probability functions is rather limited. He termed his [more realistic] view of decision-making as limited (or bounded) rationality. In this thesis, I adopt the notion that designers make boundedly rational decisions. Thus I will not attempt to derive exact measures of all parameters that may or may

not influence a designer's decisions but instead develop an understanding of the decision process itself[1]. My own research has shown that designers only have a limited understanding of how they make decisions which convinced me that a good decision model would be of great benefit to them.

IV.2.2 Optimizing Behavioral Choices

Given that all the information defined earlier is available to a decision-maker and that she has no computational limitations, I can devise different optimization strategies:

— *Expected pay-off rule*: The decision-maker maximizes the expected total pay-off by selecting the behavioral alternative with the highest expected pay-off.

$$\underset{a \in A}{Max} \sum_{w \in W_a} p_a(w) \times U(w)$$

If only subsets of behavioral alternatives and possible outcomes are considered, we get the modified decision objective:

$$\underset{a \in \bar{A}}{Max} \sum_{\omega \in \bar{\Omega}_a} p_a(\omega) \times U(\omega)$$

— *Max-min rule*: An alternative rule that is sometimes used in decision theory is the max-min rule which does not require any knowledge of

---

[1] Simon (1978) observes that economics has traditionally been concerned with *what* decisions are made rather than with *how* they are made — with substantive rationality rather than procedural rationality. In contrast, modern management science has started to fill this deficiency as it is concerned with good methods for reaching good decisions. Since this thesis is at the intersection of management science and economics, I have adopted a dual approach — (1) use decision theory to study and to prescribe *how* decisions are and ought to be made (procedural rationality); and (2) use economic reasoning to find underlying causes *why* certain decisions are made (substantive rationality).

probability $p_a(\omega)$ (and applies if computational limits are present). In this approach, the decision-maker determines the lowest possible pay-off in each behavioral alternative (minimization) and then selects the alternative for which the lowest pay-off is as large as possible.

$$\underset{a \in A}{Max} \; \underset{\omega \in \Omega_a}{Min} \; U(\omega)$$

If only subsets are considered, we get:

$$\underset{a \in \tilde{A}}{Max} \; \underset{\omega \in \tilde{\Omega}_a}{Min} \; U(\omega)$$

It is easy to see that a max-min rule is a pessimistic approach to decision-making since the decision-maker tries to optimize over the worst possible scenario. Alternatively, one could also take an optimistic position and optimize over the best of all possible outcomes which would result in a max-max rule.

Other optimization rules that have been proposed by researchers involve the use of a regret (opportunity loss) function, as opposed to utility. As decision-makers try to minimize regret, they follow rules similar to the ones described above. (For example, the max-min rule becomes a min-max rule.)

IV.2.3 Learning about Parameter Changes

In the general decision model presented, it is assumed that the decision-maker can assign some values to the decision parameters being considered. The parameter values are derived from evaluations of all available [and considered] information accumulated up to the decision point and are dependent on the decision-maker's personal judgment and

experience. In my earlier discussion, I have adopted a dynamic view of experimentation; experimental cycles are repeated iteratively until an [economically] acceptable quality level is reached by the experimenter. I will now apply the same reasoning to the presented decision model. As iterative experimentation reveals new information during each cycle, experimenters will revise their assessments of decision parameters (depicted in figure IV.2).

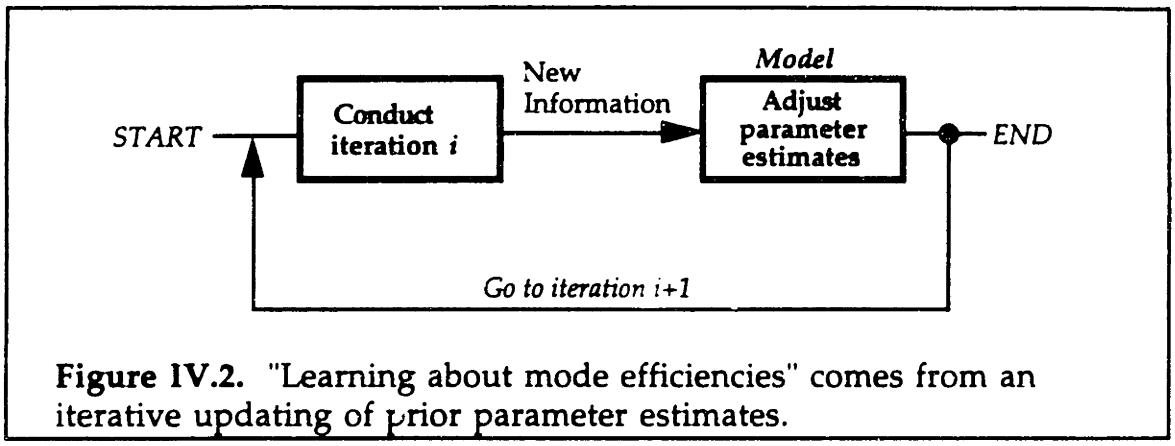


Figure IV.2. "Learning about mode efficiencies" comes from an iterative updating of prior parameter estimates.

Learning occurs because of the following reasons:

(1) *Parameter refinement under static conditions*: Under static experimentation conditions, decision parameter estimates are based on a combination of limited information and experience and therefore have levels of uncertainty associated with them. But since conditions are static, the underlying true value of the parameter remains unchanged throughout experimentation. As more experiments are conducted, additional information about decision parameters becomes available which allows the decision-maker to refine parameter estimates and the associated uncertainty.

(2) *Parameter refinement under dynamic conditions*: Very often conditions change in the course of a design project and decision parameters have to be continuously adjusted in order to reflect the underlying changes. Since it is virtually impossible to anticipate the trajectory of change prior to the start of a project, frequent information gathering by experimentation is the only reasonable way to learn about these changes.

In experimentation, I expect both, parameter refinement under static and dynamic conditions to take place simultaneously.

Mathematically, the process of iterative learning about mode efficiencies may be represented by an area of probability theory known as Bayesian analysis. In Bayesian analysis, one assumes that there is (1) a prior probability distribution which expresses *ex ante* knowledge prior to conducting an experiment; and (2) a posterior probability distribution which expresses *ex post* knowledge after an experiment is completed. For example, a semiconductor process engineer is faced with the decision whether to interfere with a single wafer film deposition process ("it is out-of-statistical control") or to let it run without interference ("it is in-statistical control"). Her estimate of the likelihood of an out-of-control process before conducting a new run — the prior probability — will be based on actual data collected from previous runs, combined with her experience and other observations that may be relevant to her decision. After her decision is carried out, the process will continue and new information becomes available from a new process run. With the new information and her estimate prior to conducting new runs, she will formulate a revised estimate of the out-of-control

likelihood — the posterior probability. Her new estimate becomes the prior estimate for the next run, and so on. In essence, she is learning from new experiments that are conducted on a run-by-run basis. If the process is stable, the new information will allow her to refine her parameter estimates. If the process is changing, the new information will reflect the underlying changes which again will allow her to adjust her parameter estimates. In either case, learning occurs by dynamically updating assessments of the decision parameters in question.

IV.3 Dynamic Experimentation Strategies in Design

In this section, I will apply the general decision model from section IV.2 to the formulation of a model that can aid the repeated (cycle-by-cycle) evaluation of mode efficiencies.

IV.3.1 Mode Switching in Experimentation

In chapter III, I have proposed that the efficiencies of repeated experimental cycles in selected modes (e.g. simulation, prototype testing) decrease at different rates. Thus, as a design progresses, designers will find it economical to determine if and where their efficiency trajectories intersect (the optimal switching point (OSP)) and revise their switching strategies accordingly (see figure IV.3 on the next page).

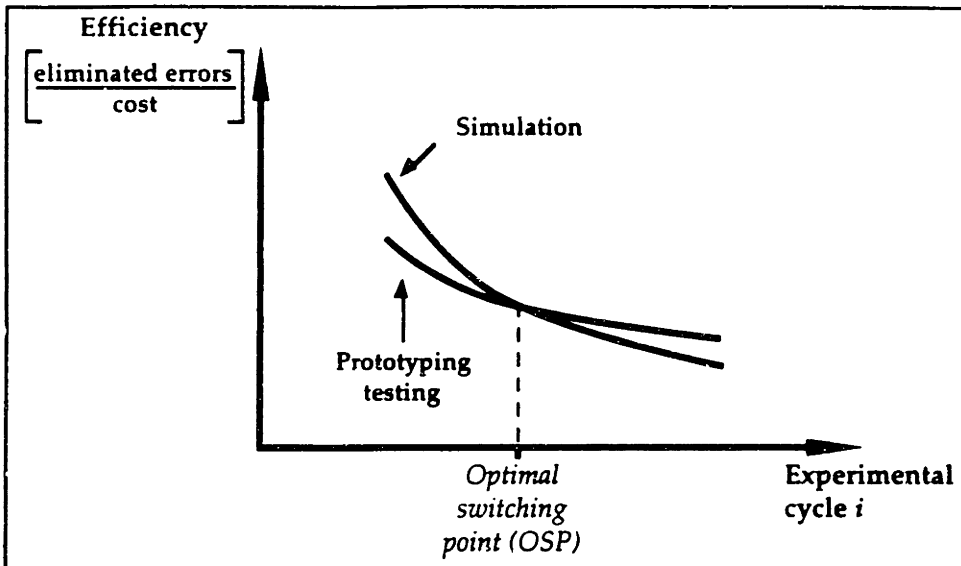


Figure IV.3. Switching from simulation to prototype testing (illustration).

Furthermore, I proposed that the modes' efficiency trajectories are unknown prior to starting experimentation and therefore designers have to learn about them from repeated experimentation. Thus, prior to starting a new experimental cycle, a designer has expectations about incremental cost, quality loss (e.g. from possible design errors), and the modeling accuracy of a given mode, and these expectations are a consequence of information gathered from previous experiments with similar designs and the designer's accumulated long-term experience with other designs.

As a result, a designer has the following choices prior to starting a new experimental cycle (see figure IV.4 on the next page):

— Discontinue experimentation because the expected incremental cost of another experimental cycle will outweigh its expected incremental benefit.

— Continue experimentation in the same mode.

— Continue experimentation in a different experimentation mode that is more efficient than the mode currently used.

Thus, if a designer determines that continued experimentation is economical, she should use the expected incremental net cost in the selection of the best experimentation mode prior to starting a new cycle.

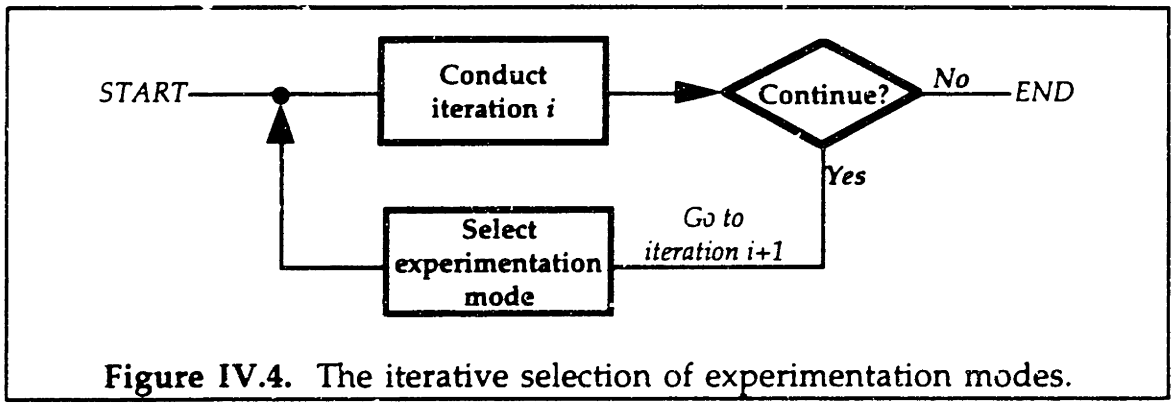


Figure IV.4. The iterative selection of experimentation modes.

IV.3.2 A Decision Model for Evaluating Mode Efficiency

We are now ready to apply the general decision model from section IV.2 to the evaluation of experimentation modes' efficiencies in eliminating design errors. (In chapter III we have established the elimination of design errors as one of the fundamental activities of experimentation and have selected to study the elimination process as part of this thesis.)

Description of Decision Variables

The general decision variables defined in section IV.2 can now be described in more detail:

— *Considered alternatives:* The alternatives that designers can choose from is the set of experimentation modes available in their design

environment (e.g. various types of simulation tools) and/or the modes that they have skills in experimenting with. These alternatives will be represented as the point set $\bar{\bar{A}}$ which consists of m modes.

— *Considered outcomes*: The set of outcomes ($\bar{\bar{\Omega}}$) if mode a is chosen can be quite large, depending on the design problem considered. As a starting point, I will consider the following possibilities:

→ If the design segment being tested has an error(s), a mode will:

↳ detect the error(s).

↳ fail to detect the error(s).

→ If the design segment being tested has no error(s), a mode will:

↳ detect no error(s).

Once the specific design is better understood, one can append more stages (as a refined list of outcomes) to each of the outcomes listed above. This will be demonstrated by example during chapter V.

— *Expected pay-off function:* The expected pay-off ($U(\omega)$) of a considered mode a consists two major components:

→ The incremental quality loss if an existing error(s) is not detected.

→ The incremental cost of conducting an experimental cycle which consists of the cost of each experimental step — build, run, analyze, and fix&verify[2].

— *Probability of outcomes:* The probability that an outcome ω occurs given that an experimentation mode a is chosen. Please note that the

---

[2] Please note that I have changed "design" to "fix&verify". In chapter III, I have explained that the *design* step consists of making changes during most of the later design stages. Thus calling the step fix&verify is a more accurate description of what designers think when making their decisions.

probabilities at each decision node (and at each stage) have to sum up to 1.

Model Discussion

A general decision model that contains the variables described above is depicted in figure IV.5 (next page). The experimentation mode is denoted as a_m, the probability of a design segment having an error is denoted as p_{a_m}, and the probability that the mode will detect an error is denoted as r_{a_m}. Similarly, the expected costs are denoted as follows: q_i is the financial quality loss and c_i^{build}, c_i^{run}, $c_i^{analyze}$, and $c_i^{fix\&verify}$ are the costs of the four experimental steps (build, run, analyze, and fix&verify, respectively).

In the decision model, each mode has three possible outcomes which are now discussed in turn:

Outcome 1: (Segment has error(s)) ∩ (Mode will detect error(s)). If both conditions are true, the expected quality loss is negligible as the detected error(s) is very likely to be eliminated. Since error elimination requires all four experimental steps (build, run, analyze, and fix&verify), the expected cost will be the sum of these steps.

Outcome 2: (Segment has error(s)) ∩ (Mode will not detect error(s)). If both conditions are true, the expected cost is the sum of the expected quality loss and the two steps — build and run.

Outcome 3: (Segment is error-free). If a segment is error-free, the expected cost is merely the sum of two experimental steps — build and run.

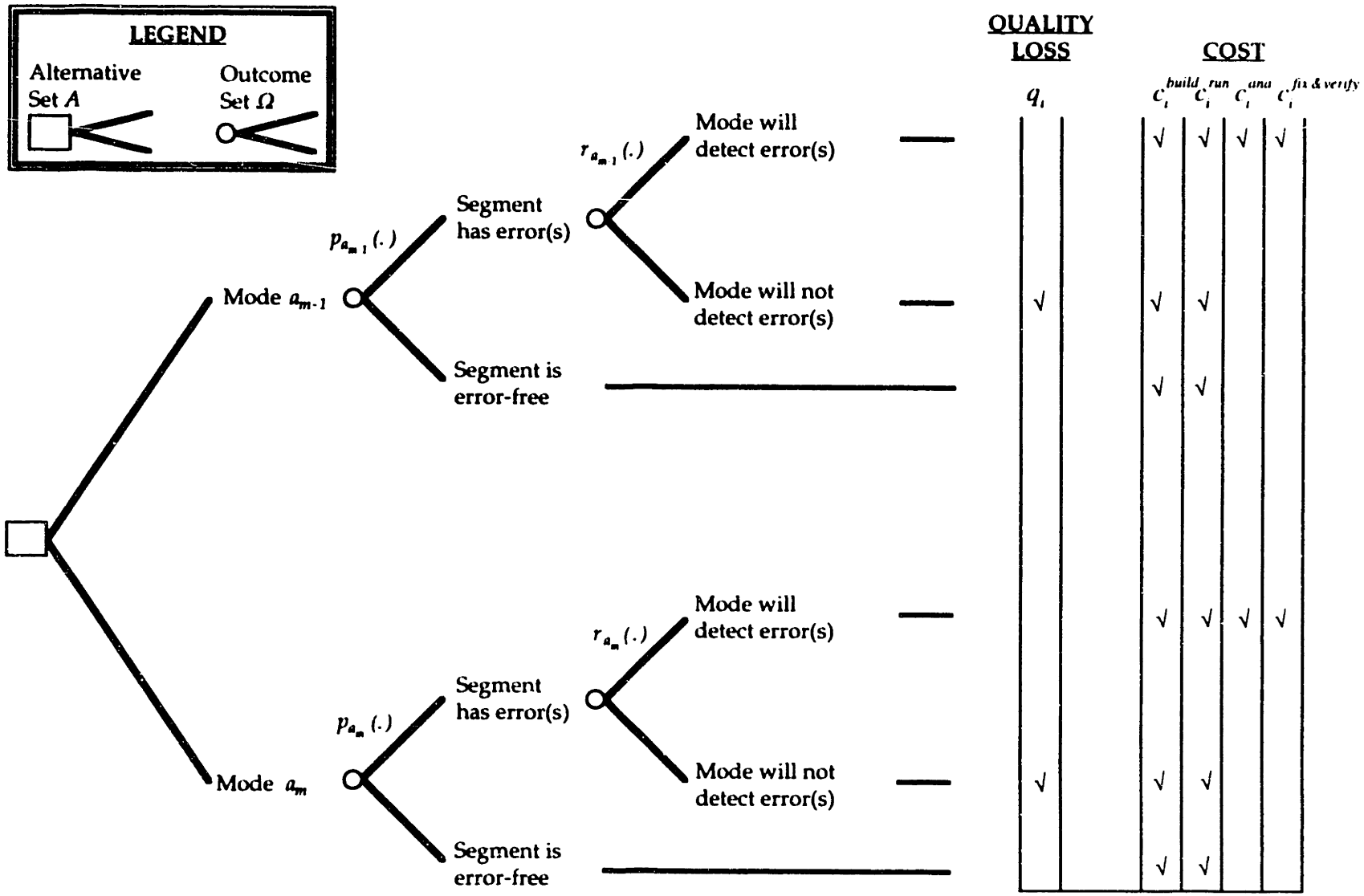


Figure IV.5. Outcome paths in selecting an experimentation mode during iteration *i*

Please note that p_{a_m} is a function of many variables such as design complexity, novelty, experience, etc. Similarly, r_{a_m} is a function of the accuracy of an experimentation model (as defined in chapter III) and test coverage. With estimates of the defined model parameters, I can now develop an expression for the expected cost $E[Cost_{a_m}]$ of an experimentation mode being considered[3]:

$$
\begin{aligned}
E[Cost_{a_m}] &= p_{a_m} \times [r_{a_m} \times (c_i^{build} + c_i^{run} + c_i^{analyze} + c_i^{fix\&verify}) + \\
&\qquad (1 - r_{a_m}) \times (q_i + c_i^{build} + c_i^{run})] + (1 - p_{a_m}) \times (c_i^{build} + c_i^{run})] \\
&= p_{a_m} \times [r_{a_m} \times (c_i^{analyze} + c_i^{fix\&verify}) + q_i - r_{a_m} \times q_i] + c_i^{build} + c_i^{run} \\
&= p_{a_m} \times r_{a_m} \times (c_i^{analyze} + c_i^{fix\&verify}) + p_{a_m} \times (1 - r_{a_m}) \times q_i + c_i^{build} + c_i^{run} \therefore
\end{aligned}
$$

(Equ. IV. 1)

I should point that some variables in equation 1 are interdependent. For example, r_{a_m} can be increased by running more test cases within an experimental cycle which again would increase the variable c_i^{run}.

Some of the parameter values in equation IV.1 change with repeated experimentation and should be re-estimated after each experimental cycle. Thus, the fundamental micro-level process of "learning about mode efficiencies" is precisely this on-going re-evaluation of model parameters.

---

[3] It is our understanding that designers rarely evaluate their options using expected values. If all model parameters can be estimated, however, a comparison of expected values is clearly the most accurate method in selecting experimentation modes.

IV.4 Conclusion

A general decision model that aids in the dynamic evaluation of a mode's efficiency (and the related mode switching) has been developed and discussed in this chapter. I proposed that this model is more efficient with respect to how designers actually decide when to switch, but at the same time simple enough to be adopted as a useful decision support tool. Specific applications and refinements of the model will be included as part of my field research in chapter V.

Chapter V

First Field Study: Grounded Research on Experimentation in Custom Circuit Design

V.1 Introduction

In this first field study, I compare computer simulation and prototype testing as experimentation modes in the design of custom circuits containing

electrically-programmable logic devices (EPLDs). More specifically, the findings of this study are used in the following contexts:

— In section V.3, I apply the four-step experimentation model to custom circuit design by examining each experimental step with respect to simulation and prototype testing. The results of the comparison are used to extend the general decision model (from chapter IV) so that it matches the mode evaluation process in custom circuit design.

— In section V.4, I report on the analysis of a data sample that consists of a detailed account of 24 design errors that were first detected during prototype testing. The findings of this analysis are used to make further refinements to the presented decision model and to derive proxy measures to test the hypotheses structure in the second field study (chapter VI). The following findings were used:

(1) In the examination of design errors detected by prototype testing, it was found that a significant proportion (46%) could not have been detected by simulation. This suggests that prototype testing is a necessary part of testing EPLD designs.

(2) Out of the errors that could not have been detected by simulation, a very significant proportion (73%) passed simulation because the simulation tools did not model the tested behavior with sufficient accuracy — a problem that affected only timing and signal quality errors[1].

---

[1] In the analysis of design errors, I divided the errors in three classes: logic errors, timing errors, and signal quality errors. These error classes differed significantly with respect to the accuracy of simulating error-related behavior (for a full explanation, see section V.4.1).

(3) Out of the errors that could have been detected by simulation, a very significant proportion (92%) passed simulation because an insufficient number of test cases was run. Presumably, designers stopped simulation "early" because, at some point, they felt that prototype testing was more effective. This result suggests that switching from simulation to prototype testing "early" (prior to the exhaustive use of simulation) is practiced by designers.

(4) Once an error is detected by prototype testing, it is very likely that the error cause will be analyzed in the laboratory (88% of examined cases). Sometimes, designers will try to recreate and analyze the error with simulation — in these findings, this occurred only in the case of logic errors (12% of examined cases).

(5) Once the error cause is found, it is likely that the cost to fix the error in EPLD designs is low (75% of examined cases). The cost of fixing an error, however, increases with error class (logic error → timing error → signal quality error).

— Finally (section V.5), I use empirical data from the field study to demonstrate how a designer can use the general decision model in the evaluation of experimentation modes.

V.2 Research Methods

This first field study had three objectives: (1) to examine information related to the [mode] switching from simulation to prototype testing, and use the findings to refine the general decision model proposed in chapter IV; (2) to apply the general decision model to a decision problem in the field; and (3)

to identify measures that can be used in studying and testing mode switching behavior in a large-scale field survey (chapter VI).

Sample Context

I elected to focus my first field study on comparing computer simulation and prototype testing in the design of custom circuits that contain electrically-programmable logic devices (EPLDs). EPLDs are high-density integrated circuits that can be quickly prototyped, tested, and modified. (EPLDs typically contain a matrix of logic elements that can be interconnected in any desired configuration to implement a given design application. For more information on EPLDs, please consult appendix B.)

In these custom designs, circuit designers make extensive use of computer simulation and/or prototype testing to detect, analyze, and correct design errors — a process that I have coined "error elimination" in the experimentation model in chapter III. The data was collected in a series of on-site interviews with designers of complex circuit design projects. The designers interviewed were employees of Bolt Beranek and Newman Inc. (BBN) in Cambridge, Massachusetts and were involved in the development of technologies for communications networking and underwater accoustics. All interviewed designers were considered as very experienced and among the best in the company.

Description of Sample

As it is often the case when circuits are developed, many design errors are detected with computer simulation but some residual errors are first encountered during prototype testing. These residual prototype errors were

the subject of my study of the efficiency of experimentation modes[2]. Thus, the sample of errors used as a study sample consisted of design errors that met the following criteria: (1) they had been first detected during prototype testing; (2) they were considered sufficiently serious to merit a design change; and (3) they were diagnosed as to cause at the time of study. Errors meeting this criteria were identified during interviews with designers by asking them to discuss design errors that were first encountered during prototype testing of a recent design. More specifically, they were asked to provide the following information: (1) a general description of the design project; followed by (2) a description of the design error; (3) the error cause; (4) the method of error detection and analysis; (5) the reason(s) for not detecting the error with computer simulation; and (6) a description of how the error was corrected. A total of 24 errors were identified and analyzed by this approach[3].

Data Collection

The data collection followed a grounded research approach (Glaser and Strauss 1970, Miles and Huberman 1984). The error data was collected over a period of three months with the aid of approximately 30 interviews, each lasting between one and two hours. All interviews were conducted on-site where respondents could consult their written and electronic records. Some

---

[2] I also tried to collect information on errors that were first detected in simulation. Designers quickly pointed out to me that they had a very poor recollection of these errors which was confirmed by my attempts to use the little information they did recall. Thus I decided to focus only on errors that were first detected in prototypes.

[3] The use of design errors and failures as a research tool has also been successfully applied in studies on software engineering. For example, see Endres (1975), Glass (1993), Selby (1990), and Thayer *et al* (1978).

interviews involved a short visit to the testing lab where physical prototypes could be inspected.

Initially, the interviewing format was nondirective but later I decided to focus on the collection of information related to specific error episodes. At the beginning of each focused interview, designers were asked to speak about design projects that involved errors that were first detected during prototype testing. All interviewees had been directly involved with these projects and were responsible for most errors recorded.

Advantages/Disadvantages of Research Methods

Conducting my research at a single company such as BBN had several advantages: the management was very supportive of my work which gave me superb access to information, the company had on-going state-of-the-art hardware development projects, and the research site was in reasonable proximity to MIT which reduced the cost of data collection significantly. At BBN, I focused my research efforts on a group of four very experienced designers that were involved in the development of sophisticated systems hardware for communications networks. All designers had experience and on-going designs that involved EPLDs. A thorough study of these designs helped me in developing a deep understanding of the micro-level mechanisms that drive the use of simulation and prototype testing in custom circuit design.

On the other hand, my research approach also had a few disadvantages:

— My interviews were with a preselected group of very experienced designers who worked for the same company and sometimes on the same projects.

— The collected error data came from a small group of designers (22 out of the 24 error cases where provided by three designers) and a small number of projects.

— For some errors, there was a significant difference between the interview date and the time the error was made. Thus, it was sometimes difficult to recreate the particular circumstances that led to the error being discussed.

— The errors (and the stories behind them) were mostly qualitative in nature and were not suitable for rigorous statistical testing.

In summary, I traded the richness of and the learning from in-depth interviews for the ability to perform statistical tests on large-size, randomly collected data samples. As a result, I performed a second large-scale field study (chapter VI) which was precisely aimed at overcoming the deficiencies from this first field study.

V.3 Experimental Cycles in Circuit Design: Comparing Simulation and Prototype Testing

The purpose of this section is to apply the model of a four-step experimental cycle to custom circuit design, and to compare computer simulation and prototype testing by examining the contents of each step. The comparison will achieve (1) an understanding of simulation and prototype testing in circuit design which is necessary for the rest of the thesis; and (2) a

first-level refinement of the general decision model of mode switching that was proposed in chapter IV.

Before starting the discussion, I should also mention that, in addition to simulation and prototype testing, some designers utilize a third experimentation mode that could be best described as paper simulations. Instead of generating a computer-based numerical model of circuit behavior, designers rely on models that remain inside their "heads". During paper simulations, waveforms of incoming signals are drawn on paper, followed by output signals that are approximated by the designer. While paper simulations are normally less accurate than computer simulations or prototype testing, they can be quite effective if used by very skilled designers because of the relatively low set-up time. Thus we often see "paper simulations" during the early phases of a design.

I will now provide a step-by-step description of an experimental cycle, comparing simulation and prototype testing at each step. The discussion is based on information that was obtained from the initial set of interviews with designers.

Step 1: Design

During *design*, circuit designers use design specifications, the design problem formulation and their own experiences to generate design solutions. The design solution is either constructed with the aid of computer-aided design (CAD) software (known as "schematic capture") or generated with the aid of high order synthesis tools. As a design progresses, activities during *design* are normally related to design changes as to fix errors found during a

previous cycle or as to increase design performance. All design modifications have to be made on software first before proceeding to build a test model (this applies to prototype testing and simulation) .

Step 2: Build

In preparing a model used by computer simulation, designers typically do the following:

(1) *Create a model of the design segment to be simulated.* This is often straightforward as all designs were entered in either schematic capture or some higher language synthesis files. These files are directly converted into a circuit description which simulation tools use to map the design on built-in numerical models. However, numerical simulation models of the design vary in accuracy with respect to the behavior being modeled (which will be discussed in detail later). If board-level simulations are planned, designers often buy numerical models for standard components (e.g. microprocessors) that can be directly "plugged in" a simulation model.

(2) *Create a model of the environment external to the design segment to be simulated.* If the external environment is another design segment that is already completed and available in electronic format, it can be directly "plugged" into the simulation model. In most cases, however, the external environment is not completed and only available in the form of interface specifications. Thus designers have to enter a complete description of the anticipated environment in a behavioral language format which can take many hours. Since knowledge about the

environment is often incomplete, so will be the external model and the associated testing by simulation[4].

In preparing a hardware model used during prototype testing, designers typically do the following:

(1) *Map the design on a set of EPLDs.* The design to be tested is "mapped" on EPLDs using software provided by the EPLD manufacturer (the process is known as "routing"). After the software finishes routing and confirms a successful fit, the design is ready to be prototyped. Depending on the EPLD technology, the complexity of the design and the software platform available, the time to route can vary significantly (in most cases, however, it takes less one hour per EPLD).

(2) *Build the prototype.* Many designs involve several EPLDs and other components that are mounted on a fabricated hardware board. Fabricating the first hardware board can be quite costly (not as costly as fabricating an ASIC, however), but design changes to it can be made quite easily as long as they remain within an EPLD (as opposed to board level changes which tend to be costly). Programming an EPLD usually takes a few minutes[5]. Thus the incremental cost of modifying a prototype tends to be relatively low for most design errors (this issue will be discussed in more detail later).

---

[4] In some cases, the description of the environment may be incomplete for proprietary reasons and the utility of simulation will be small. In such a case, the only viable strategy is iterative trial and error by prototype testing. For an example, see error case AK-2 (appendix C).

[5] If the EPLD has to be physically removed from the hardware board, it may actually take a few hours because of the difficulty in unsoldering surface mounted devices (most EPLDs are surface mounted).

Step 3: Run

In running a computer simulation, designers typically do the following:

(1) *Select a set of test vectors to be simulated.* As exhaustive simulations are not economical due to the slow speed of simulations relative to "real" prototypes[6], a representative set of test vectors has to be selected as to achieve optimal test coverage.

(2) *Initialize input signals.* Prior to starting a computer simulation, the designer has to initialize the state of all design inputs which can take several hours.

(3) *Run selected test cases.* Once a set of test vectors is selected and all inputs are initialized, the designer can instruct the simulation to be run. Simulation models are based on finite element approximations which run much slower than "real" prototypes.

(4) *Screen results for errors.* After running the test cases, the results have to screened for errors. The results are available as a sequence of waveforms which can be matched against an expected (or specified) pattern. Any deviations from this pattern will be analyzed as potential design errors.

I should also mention that sometimes designers can write software routines that automatically go through (1) to (4). However, in the group

---

[6] Even with the fastest computers available, simulation is several orders of magnitude slower than a real prototype. As a practical matter, most simulations rarely simulate real time behavior beyond the micro- to millisecond range (which can take hours or days to simulate). One may be tempted to conclude that the difference will decrease significantly with increasingly powerful computers. That is unlikely for two reasons: (1) the magnitude of the difference is very large (several orders of magnitude); and (2) the clock speed of EPLDs and ASICs is increasing quite rapidly which will offset any gains against from running faster tests.

interviewed, only one designer developed software routines and only used them in testing a subset of all error-related behavior[7].

In running prototype tests, designers typically do the following:

(1) *Run diagnostics.* Similar to selecting test vectors in simulation, designers have to select a set of diagnostic laboratory tests that cover most of the expected operational use of the design. Running selected routines allow the designer to isolate different aspects of a design which, if an error occurs, significantly reduces the search space for the error cause (experimental step (4)). Because prototype testing can be run in real-time, the actual time required to run tests is usually small.

(2) *Run in full operation.* In full operation, the design runs as if it was operating in the field. This is especially important in the detection of stochastic errors (i.e. errors that are triggered by random events occurring after minutes or hours of full speed operation). Stochastic errors cannot be detected with simulation because of the described run-time constraint.

Step 4: Analyze

In analyzing test results with the aid of simulation, a designer typically enjoys many advantages. Some of them are:

---

[7] Not all behavior is suitable for software automation. For example, consider the following types of logic behavior in digital circuitry:

(1) *Combinatorial logic:* the number of cases to be simulated is determined by the number of inputs. Thus the total number of cases = 2^{inputs}. Running combinatorial logic simulation can be automated with software code and the test coverage is typically very high.

(2) *Sequential logic:* the number of cases to be simulated is determined by the number of inputs and the number of states. Thus the total number of cases = $2^{inputs} \times 2^{states}$ which is a very large number even for small finite state machines. Running sequential logic simulations with automated software code is usually very difficult to implement.

— Signals can be analyzed at arbitrary nodes inside a design which enables a designer to trace a signal through all internal circuit nodes between design inputs and outputs.

— Multiple signals can be displayed simultaneously and executed with varying degrees of clock speed (e.g. clock-by-clock). This allows a search for unusual patterns in the behavior of the circuit component simulated.

— Small design parts can be resimulated in isolation which allows a more systematic problem solving approach in finding an error cause.

In analyzing test results using prototype testing, a designer typically encounters several difficulties. The most frequently mentioned are:

— It is often impossible to access signals via nodes that are internal to the EPLD. While some EPLD vendors provide limited access to internal nodes with the help of built-in probing firmware, the degree of access to internal signals is by no means comparable to simulation.

— The analysis is normally done with the aid of oscilloscopes and logic analyzers. In order to reduce the problem space to a manageable size, one has to start by finding a "triggerpoint" (i.e. find the point in the time domain when the problem occurred). Very often, searching for a triggerpoint can be quite difficult and requires a substantial skill level.

— Probing input and output pins of an EPLD is quite difficult because of the small size of these pins. In fact, they are so small that designers (or test engineers) often need magnification glasses to see the spacing between pins. With decreasing package sizes and increasing pin counts, probing is expected to get even more difficult in the future. Unfortunately,

available oscilloscope or logic analyzer probes are not designed to effectively deal with this problem.

In summary, simulation tends to be a lot more effective in the analysis of an error if it is accurate in modeling error-related behavior. Since designers often do not know in advance if an error detected by prototype testing is due to a lack of simulation accuracy (i.e. simulation cannot aid in the error analysis) or an insufficient number of test cases simulated (i.e. simulation can aid in the error analysis), they sometimes try to recreate the error in computer simulation after a triggerpoint has been found in the lab.

Refining the General Decision Model for Circuit Design

With the information presented, one can refine the general decision model from chapter IV by appending two additional stages to the model (see figure V.1). These two stages correspond to the two experimental steps (4) (analyze) and (1) (fix error) which were not explicitly[8] referenced in the general decision model's tree structure. This new model enables designers to assign measures of likelihood to steps (4) and (1) and therefore improve the general accuracy of the decision model.

[8] The two steps were referenced implicitly by measuring their cost.

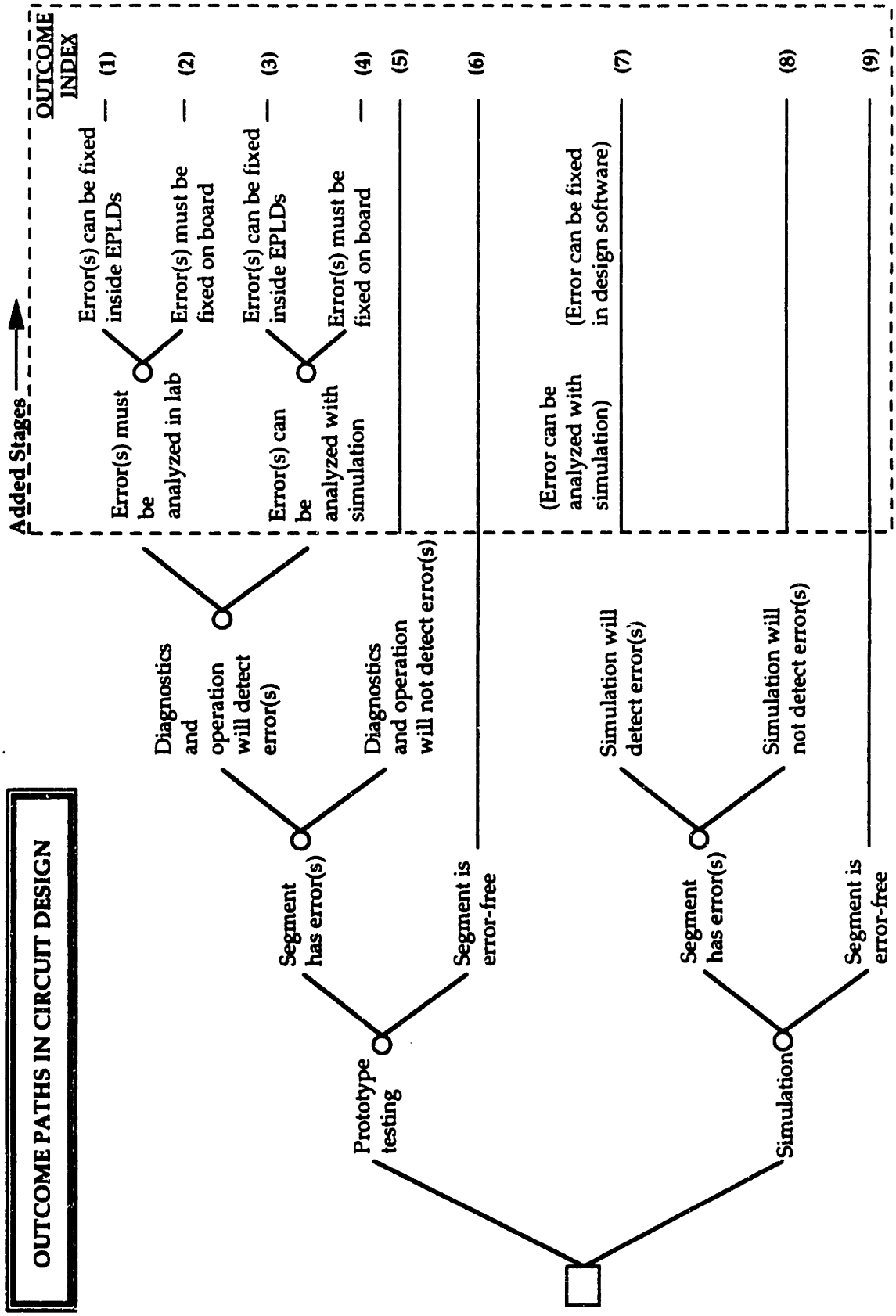


Figure V.1. Outcome paths related to the switching between computer simulation and prototyping.

The **first** added decision stage addresses the analysis of errors (step 4). With respect to the analysis of errors detected by prototype testing, the prior discussion has shown that designers normally have two options: (1) find the error cause in a laboratory with the aid of oscilloscopes and logic analyzers; or (2) recreate and analyze the error in computer simulation. Sometimes designers use a mixed approach; they start in the lab and then switch to a simulation and back. Unfortunately, option (2) only works if the designer has a triggerpoint (i.e. knows where in the time spectrum the error has occurred) and if the simulation is accurate enough to detect the error. With respect to the analysis of errors detected by simulation, detecting an error by simulation implies that the error can be analyzed with the aid of simulation.

The **second** added decision stage addresses the fixing of errors (step (1)). With respect to fixing an error detected by prototype testing, field interviews and inspections of hardware designs suggested the following two possibilities: (1) errors can be fixed by making a design change in the design software and reprogram an EPLD (low-cost solution)[9]; or (2) errors have to be fixed on the hardware board (which can be a high-cost solution — especially for designs that will be manufactured in high volume[10]). With respect to fixing an error detected by simulation, errors can be fixed directly on the design software.

---

[9] The EPLD is removed from the board and replaced by a newly programmed EPLD. Some vendors offer EPLDs that can be reprogrammed on board and thus do not require the tedious removal of soldered ICs.

[10] Sometimes errors can be fixed on-board with the aid of "blue wires" which connect various points on the board layout. The use of blue wires avoids a costly redesign and respinning of the hardware board which can cost up to a few thousand dollars for very large designs. However, blue-wires are only practical for low-volume production (<20 boards) and are not always feasible.

V.4 Findings

In this section I present the collected error data and analyze it in the context of the general experimental model. More specifically, I intend to use the data for the following purposes: (1) to further refine the decision model for mode switching that was presented in figure V.1; (2) to identify parameters that can be used to measure hypotheses-related effects during the second field study (chapter VI); and (3) to find some initial empirical results that underpin the hypothesized differences between simulation and prototype testing.

V.4.1 Description of Data

Data on 24 design errors that passed computer simulation and were first detected during prototype testing was collected as part of my interviewing process. Out of all errors reported, 21 were made by the interviewed designers themselves and 3 were made by junior designers who worked directly with the designer who reported the error. A break-down of the error data is shown in table V.2 and a summary of the errors is provided in tables V.1a and V.1b. (which are on the following two pages; for more details on these errors, please consult appendix C).

Code	Design Project	Brief Description of Error	Class
AK-1	Long link emulator (error generator chip).	Made change to circuit and didn't resimulate (calculated decision because resimulation would have taken considerable time). Forgot bubble (missing negation on logic input).	L
AK-2	Serial protocol converter (connect PictureTel format and HDLC packet format).	Assumption about protocol was incorrect (CRC-cyclical redundancy check) and not available in data books. Couldn't simulate. Solution was found by 30-40 prototype iterations.	L
LD-1	Design of main board and 4 daughter boards for an internet router (BI4).	Two select lines on 2-to-4 decoder were accidentally switched; simulated only first case (the remaining 3 were identical). Weighed incremental effort of simulation against risk-adjusted incremental effort to detect and fix.	L
LD-2	Design of main board and 4 daughter boards for an internet router (BI4).	Timing problem that occurred only every 20 minutes (68020 needed access to ethernet) of real time execution. Could only simulate reasonably if problem and cause known a priori (i.e. exhaustive simulation impossible).	T
LD-3	Design of main board and 4 daughter boards for an internet router (BI4).	Flash memory couldn't drive slow bus. Incorrect assumptions during design (it was reasonable to assume that memory could drive bus). Couldn't simulate because of (1) line capacitance; and (2) incorrect assumption.	T
LD-4	Design of hydrophonic data acquisition card (LMSHSU).	The +5V ground planes were not partitioned which resulted in noise problem during A/D conversion. Cannot be simulated.	S
LD-5	Design of hydrophonic data acquisition card (LMSHSU).	The point at which the acquisition board samples the A/D converter was displaced in time. Poor sim. tools & schedule pressure made sim. difficult.	L
LD-6	Design of hydrophonic data acquisition card (LMSHSU).	Left mode pin unassigned which unexpectedly resulted in probing mode. Couldn't simulate probing feature (EPLD manufacturer specific). Assumed that unassigned pin results in "no probe".	L
LD-7	Design of hydrophonic data acquisition card (LMSHSU).	Mixed up MSB and LSB on interrupt vector. Couldn't simulate because assumption was built in simulation (LD wrote the VME bus simulation) .	L
LD-8	Switch card for networking box (Emerald project).	One of the select lines on the decoder was inverted. Oversight. Didn't simulate because (1) didn't expect error; and (2) EPLD wasn't full (easy fix).	L

Notes: The error classes are: L = logic errors; T = timing errors; S = signal quality errors.

Table V.1a. List of errors (for a full description of errors, see appendix C)

Code	Design Project	Brief Description of Error	Class
PC-1	Upgrade design of network processor (ACTS).	Unclocked flip flop set off by random transients. Could have been prevented by cautious design approach (synchronous). Simulation couldn't detect it.	T
PC-2	Burst modem interface for ACTS.	One op. amp (in the analog section) had its inputs reversed (inverting and non-inverting inputs were switched). Did not simulate analog section.	L
PC-3	Burst modem interface for ACTS.	Two lines on power supply made contact on main board. Cause unknown. Couldn't simulate (error probably occurred during board design).	S
PC-4	Burst modem interface for ACTS.	PC used expander latch to design Tx offset register. Solution was asynchronous and didn't work. Couldn't simulate subtle timing variation.	T
PC-5	Power supply for burst modem (ACTS).	Designer forgot to insert capacitors for current peaks (she didn't understand the consequences of missing capacitors). Couldn't simulate analog behavior.	S
PC-6	Terminal controller for ACTS.	Ran simulation on test formula to economize on test cases. Used wrong sign in formula and missed error. Could have been caught by simulation.	L
RP-1	Queued interrupt controller for ACTS.	Error caused the wrong bit to be cleared. To economize on simulation, only 1-2 cases were tested. Could have been caught by full simulation.	L
RP-2	Queued interrupt controller for ACTS.	Bus arbiter rule gave WRITE priority over READ. Under heavy traffic conditions, READ couldn't access bus. Could have been simulated if the right test conditions had been selected.	L
RP-3	Bus interface chip for ACTS.	Arbiter didn't have enough time to decode address. Time was on the margin, passed simulator because pin capacitance & line impedance aren't modelled.	T
RP-4	Bus arbiter for ACTS.	Assumption about burst transfer rate was incorrect. Made change to module but didn't resimulate (change probably resulted in change of burst rate).	L
RP-5	RAM controller for comm. satellite (ACTS).	Forgot to resimulate extra feature (usage of padding bits) after specification change occurred. Zero values weren't allowed anymore which led to error.	L
RP-6	Bus interface chip for ACTS.	Didn't simulate counter value 0 which resulted in a no-frame pulse. Error could have been caught by simulation but simulated only partial counter.	L
RP-7	SONET line card for ACTS.	Pulse for location identification in data stream was two clock cycles off. Didn't simulate interface because it would have required new behavioral model.	L
RP-8	SONET line card for ACTS.	Address sequencer didn't work right. Thought it was microcode problem. Did not simulate for entire 4ms (only the first 0.25ms). Error occurred after 0.75ms.	L

Notes: The error classes are: **L** = logic errors; **T** = timing errors; **S** = signal quality errors.

Table V.1b. List of errors continued (for a full description of errors, see appendix C)

Error was reported by:	Logic Errors	Timing Errors	Signal Quality	Total
(1) Designer AK	2	0	0	2
(2) Designer LD	5	2	1	8
(3) Designer PC	2	2	2	6
(4) Designer RP	7	1	0	8
Total	16	5	3	24

Table V.2. Summary of reported errors (by error class and coded name of designer).

The reader may note that I have divided the errors in three classes: logic errors, timing errors, and signal quality errors. The reasons for classifying errors into these categories were: (1) circuit designers often use the same classification when making decisions with respect to switching from simulation to prototype testing; (2) simulation tools are typically designed to deal with each of these error classes separately; (3) an initial examination suggests that the difficulty of modeling error-related behavior varies significantly between the three classes which may provide me with a proxy for measuring "modeling accuracy" (related to hypothesis 3). To explain the error class distinction, I will now discuss each of them in turn.

Error Class 1: *Logic Errors*

Logic errors relate to the functional behavior of a circuit. In digital circuits, functional behavior is represented by binary logic which is an idealized representation of "real" behavior. During logic simulation, variations in the physical world (such as noise, delays, etc.) are usually ignored. Thus, with the aid of mathematical tools such as Boolean algebra,

logic behavior can be modeled with great accuracy. Therefore simulation can almost always detect logic errors if a test case simulated the error-related behavior. An example of a logic error is shown in figure V.2 (source: appendix C).

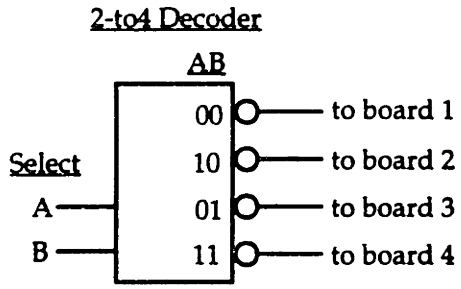


Figure V.2. *Example of a logic error*: In the design of a 2-to-4 decoder, the select lines A (least significant bit) and B (most significant bit) were accidentally switched. As a result, the addresses for board 2 and 3 were reversed.

Error Class 2: Timing Errors:

Timing errors relate to the timing behavior of a circuit. As signals switch and propagate through a physical circuit, they typically encounter delays due to capacitive and resistive loading of interconnects and circuit cells. These delays influence the speed at which signals can rise and fall and thus affect overall predictability of timing performance, especially during transient states. Timing simulation often uses propagation delay models which are usually [first or second order) approximations of real behavior and depend on the physical layout of a design. The accuracy of these timing models varies but they are significantly less accurate than models of logic behavior. Thus timing errors may go undetected during timing simulation,

even if the appropriate test cases are simulated. An example of a timing error is shown in figure V.3 (source: appendix C).

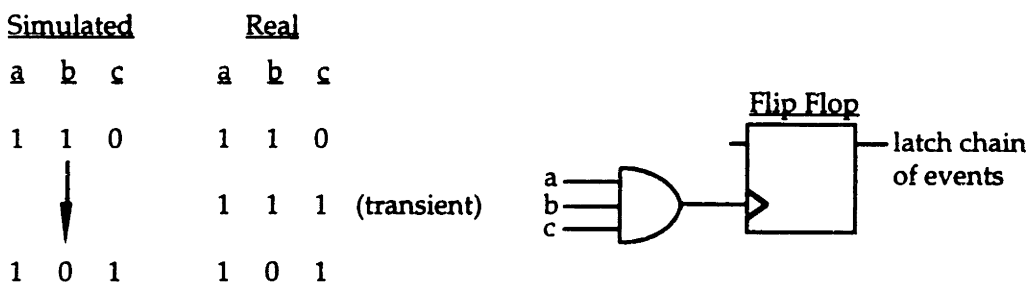


Figure V.3. *Example of a timing error*: In the design of a latch, the latch was triggered by a flip-flop if statebits a, b, and c were 111. During normal timing operations, random transients occurred during statebit transitions. Because of complex timing delays, it was impossible to predict these transients (or simulate them) and therefore the designer should have clocked the design configuration (i.e. use synchronous timing). In this case, the designer failed to clock the design and the latch was set by random transients approximately once per two hours (see "real" versus "simulated" transition in picture).

Error Class 3: Signal Quality Errors

Signal quality errors relate to the deterioration of signal quality (or integrity) as a signal propagates through a physical circuit, and these errors are highly dependent on the physical hardware being used. Signal quality usually decreases due to the introduction of noise which can come from many sources. Examples are: (1) proximity cross-talk of two interconnects in high-speed circuits; (2) on-board components that radiate noise; and (3) poor shielding from noise sources external to the board. Since noise is difficult to simulate (normally statistical approximations are used) and its effect on signal quality extremely difficult to model, few designers rely on signal quality

simulations. (The simulation of signal quality-related behavior is significantly less accurate than timing simulation, and much less accurate than logic simulation.) Instead, designers try to follow design rules that result in a higher noise robustness of circuits. Thus, signal quality errors are rarely detected in simulation. Finally, I should mention that signal quality errors are more frequent in mixed signal designs where analog and digital signals interact. An example of a signal quality error is shown in figure V.4 (source: appendix C).

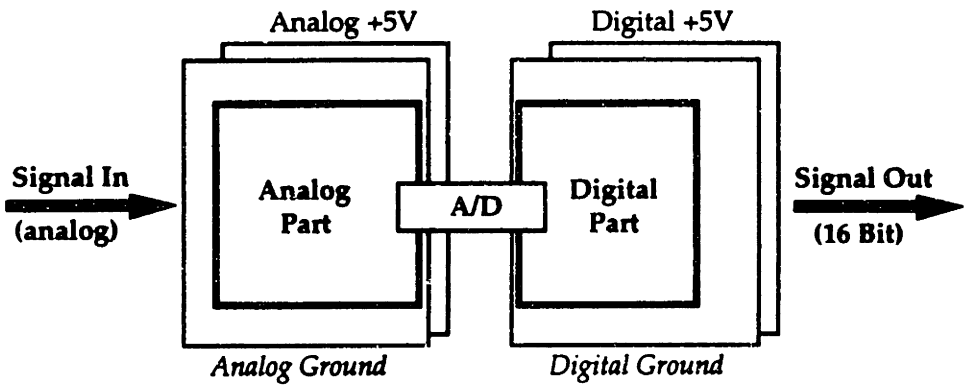


Figure V.4. *Example of a signal quality error*: In the design of a custom analog-to-digital converter, the designer failed to separate the +5V plane between the analog and the digital parts. As a result, the signal quality was degraded by noise that spilled from the +5V plane and the digital output was offset by several bits.

V.4.2 The Detection of Design Errors

I start the data analysis by comparing the efficiency of simulation and prototype testing in the detection of design errors (experimental step (3) (run)). The first issue I was interested in determining was whether prototype testing is necessary in the elimination of design errors or whether all errors could have been detected by simulation. I did this by dividing the total

sample of errors in two groups: (1) errors that could not have been detected by simulation; and (2) errors that could have been detected by simulation but were not. If there was a significant proportion of errors that could not have been detected by simulation, it would support the proposition that the efficiency of simulation has limitations, and that prototype testing is necessary to go beyond these limitations. An analysis of the errors is presented in table V.3.

Why was the error not detected with simulation ?	**Logic Errors**	**Timing Errors**	**Signal Quality**	**Total**
(1) Simulation could not have detected the error. — *Example*: In the design of a bus interface chip, the bus arbiter did not have enough time to decode the bus address. However, simulation indicated that there was sufficient time because the timing model did not consider timing delays due to line impedance.	3	5	3	11
(2) Simulation could have detected the error. — *Example*: In the design of an INTERNET router, two select lines of a 2-to-4 decoder were accidentally switched. An exhaustive simulation of the decoder would have required 4 very similar test cases (one per decoded address). Because the designer tried to economize on simulation, he simulated only one test case which passed simulation (two of the four test cases would have failed) and assumed the rest of the design to be correct.	13	0	0	13
Total	16	5	3	24

Table V.3

The data indicates that 46% (11) of all errors could not have been detected by simulation which implies that getting the prototype "right the

first time" would have been a nearly impossible task with the simulation tools that were available to the designers at the time of testing.

Result 1: In the examination of design errors detected by prototype testing, it was found that a significant proportion (46%) could not have been detected by simulation .

Next, I examined the reasons why simulations could not have detected the group of 11 errors. This was done by dividing them into two categories: (1) errors that went undetected because the simulation tools did not model the tested behavior with sufficient accuracy; and (2) other reasons. If accuracy was a significant reason, it would suggest that accuracy has an effect on simulation efficiency and thus could be used in the testing of hypothesis (3) (see hypotheses structure in chapter III). As can be seen in table V.4, 73% (8) of these design errors could not be detected by simulation because of insufficient accuracy. Furthermore, insufficient accuracy had an impact on timing and signal quality errors but did not affect logic errors — a finding that is consistent with the difficulty of modeling the respective error-related behavior.

(1) Simulation could not have detected the error...	Logic Errors	Timing Errors	Signal Quality	Total
(a) because the simulation tool did not model the tested behavior with sufficient accuracy.	0	5	3	8
(b) other reasons (e.g. simulation tool lacks capability).	3	0	0	3
Total	3	5	3	11

Table V.4

Result 2: Out of the errors that could not have been detected by simulation, a very significant proportion (73%) passed simulation because the simulation tools did not model the tested behavior with sufficient accuracy — a problem that affected only timing and signal quality errors.

Another important aspect of this finding is that the different simulation tools can be regarded as a family of simulation "modes" that differ significantly in the accuracy of their respective models. In other words, because of the difference in accuracy, their efficiency trajectories are likely to be different — with logic simulation to be most efficient and signal quality simulation to be the least efficient (see figure V.5). Thus, the different simulation tools can be used in the second field study (chapter VI) as proxies for variations in accuracy which would allow me to test hypothesis (3).

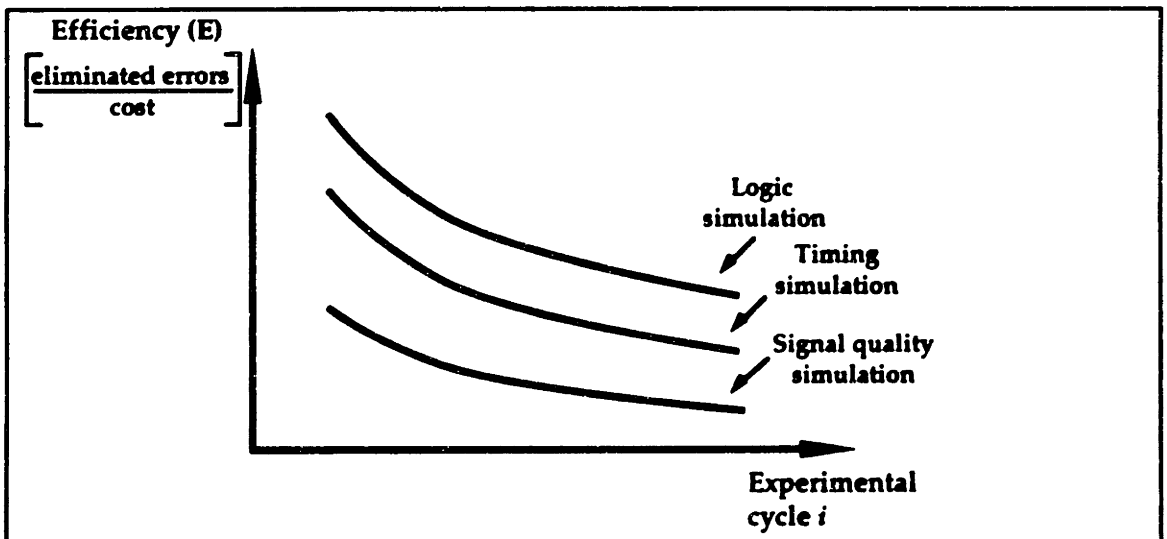


Figure V.5. Efficiency differences between different simulation tools (illustration).

Next, I examined the reasons why simulations could have detected the group of 13 errors but failed to do so. Again, this was done by dividing them into two categories: (1) errors that went undetected because not enough test cases were run (i.e. designers switched to prototypes "early"); and (2) other reasons. If it was found that a significant proportion of errors could have been found by running more test cases in simulation but designers decided not to do so, it would support that, at some point during repeated experimentation, they found prototype testing to be more economical (which relates to the concept of mode switching proposed in chapter III). Table V.5 shows that 92% (12) of the errors that could have been detected by simulation, were not detected because not enough test cases were run. Thus, there is some preliminary evidence that designers find it economical to switch to prototype testing before simulation has detected all errors it could detect.

(2) Simulation could have detected the error...	Logic Errors	Timing Errors	Signal Quality	Total
(a) if additional test cases had been simulated (i.e. needed higher test coverage).	12	0	0	12
(b) other reasons (e.g. designer supplied simulation tool with incorrect information).	1	0	0	1
Total	13	0	0	13

Table V.5

Result 3: Out of the errors that could have been detected by simulation, a very significant proportion (92%) passed simulation because an insufficient number of test cases was run. Presumably, designers stopped simulation "early" because, at some point, they felt that prototype testing was more effective.

V.4.3 The Analysis of Design Errors

In the earlier discussion on experimental cycles in circuit design, I have proposed that the analysis of errors (experimental step (4)) (1) is performed with aid of simulation if simulation detected an error; or (2) is performed in the laboratory (and sometimes with the aid of simulation) if an error is detected in prototype testing. Since all the collected data consists of errors that were detected by prototype testing, I will examine whether the findings are consistent with this earlier proposition, and if one can make further refinements to it. As can be seen in table V.6, 88% (21) of all errors were analyzed in the laboratory and 12% (3) were analyzed with the aid of simulation. Furthermore, all timing and signal quality errors were analyzed in the laboratory and some logic errors were analyzed with the aid of simulation.

How did you find the error cause(s) ?	Logic Errors	Timing Errors	Signal Quality	Total
(1) The error cause(s) was analyzed in the laboratory. — *Example*: The designer used lab equipment (e.g. oscilloscope, logic analyzer) and the design schematic to find the error cause.	13	5	3	21
(2) The error cause(s) was analyzed with the aid of computer simulation. — *Example*: The designer went back to his office and tried to recreate the error [symptom] with the aid of computer simulation. If successful, he could have analyzed the error cause with simulation.	3	0	0	3
Total	16	5	3	24

Table V.6

Result 4: Once an error is detected by prototype testing, it is very likely that the error cause will be analyzed in the laboratory (88% of examined cases). Sometimes, designers will try to recreate and analyze the error with simulation — in these findings, this occurred only in the case of logic errors (12% of examined cases).

V.4.4 The Correction of Design Errors

Next, I examined the cost of correcting (or fixing) design errors after their cause had been found (experimental step (1)). Earlier I suggested (1) that errors detected and analyzed with the aid of simulation can be corrected at very low cost; and (2) that errors detected and analyzed by prototype testing can be corrected at low cost (if fixed inside EPLDs) or high cost (if fixed outside EPLDs). Since the research sample applies to case (2), I divided the data in three categories: errors that could be fixed easily (low cost), errors that could be fixed with moderate difficulty (medium cost), and errors that could be fixed only with much difficulty (high cost).

As can be seen in table V.7 (next page), 75% of all errors detected by prototype testing could be fixed easily. A closer examination also shows that 94% (15) of all logic errors could be fixed easily, whereas timing errors could be fixed either easily (60% – 3 errors) or with moderate difficulty (40% – 2 errors). In contrast, all 3 signal quality errors could only be fixed with much difficulty (100%).

Result 5: Once the error cause is found, it is likely that the cost to fix the error in EPLD designs is low (75% of examined cases). The cost of fixing an error, however, increases with error class (logic error → timing error → signal quality error).

How difficult was it to fix the error ?	Logic Errors	Timing Errors	Signal Quality	Total
(1) Fixing the error was easy. — *Example*: Fixing the error required (a) the removal of an EPLD from the board; (b) a minor design modification; (c) the reprogramming of a new EPLD; (d) the resoldering of the new EPLD.	15	3	0	18
(3) Fixing the error was somewhat difficult. — *Example*: Fixing the error required (a) the removal of at least one EPLD from the board; (b) several design modifications; (c) the reprogramming of at least one EPLD; (d) the resoldering of the new EPLD(s).	1	2	0	3
(3) Fixing the error was very difficult. — *Example*: Fixing the error required physical changes to the board (e.g. the board had to be redesigned).	0	0	3	3
Total	16	5	3	24

Table V.7

V.4.5 Applying the Findings to a Refinement of the Decision Model

With the findings in the previous section, one can make further refinements to the decision model (figure V.1) used in evaluating the expected efficiency of experimentation modes.

In summary, the following results can be included:

Detect:

— *Simulation*: The accuracy of simulation models decreases with error class (logic → timing → signal quality). Thus, simulation is very accurate for logic errors but is very inaccurate for simulating signal quality-related behavior. (Supported by result 2 and background information.)

— *Prototype testing*: If a complete prototype is built, prototype testing is very accurate by definition.

Analysis:

— *Simulation:* Errors that are detected by computer simulation can usually be analyzed in simulation and fixed in the design software.

— *Prototype testing:* (a) Logic errors that are detected by prototype testing can sometimes be analyzed with simulation; (b) timing errors that are detected by prototype testing are usually analyzed in the lab; and (c) signal quality errors that are detected by prototype testing are usually analyzed in the lab. (Supported by result 4 and confirmed in follow-up interviews.)

Fix Errors:

— Simulation: Errors that are detected and analyzed by simulation are normally fixed in the design software (very low cost).

— Prototype testing: (a) Logic errors that are detected and analyzed by prototype testing can usually be fixed inside an EPLD and do not require board-level changes (low cost); (b) timing errors that are detected and analyzed by prototype testing can be fixed either inside an EPLD or may require more difficult changes that involve multiple EPLDs (medium cost); and (c) signal quality errors that are detected and analyzed by prototype testing must usually be fixed at the board level (high cost). (Supported by result 5 and confirmed in follow-up interviews.)

Including these results in the decision model can greatly simplify the assessment of likely outcomes since it enables the designer to follow

dominant outcome paths. Decision models with such dominant outcome paths as a function of error class are shown in figure V.6 (for logic errors), figure V.7 (for timing errors), and figure V.8 (for signal quality errors) (on the following pages).

Please note that the depicted likelihood of detecting an existing error assumes perfect test coverage (figures V.7, V.8, and V.9). However, we have seen earlier that a designer may deliberately decrease test coverage because the incremental cost of achieving higher test coverage exceeds the probability adjusted quality penalty of an undetected error. In such cases, the likelihood of detection must be decreased.

V.5 Using the Decision Model to Evaluate Experimentation Modes: An Empirical Example

In chapter IV, I developed a general decision model for the selection of experimentation modes in design. In this section, I will use data that was provided by one of the interviewed designers to demonstrate how the model can be used in a decision problem. As mentioned earlier, using a decision model will prove to be very helpful to designers since, based on my observations, they usually do not conduct systematic analyses before a mode is selected. Instead, they rely on simple heuristics that evolve with design experience [11].

---

[11] In all fairness, it should be pointed that circuit designers are probably more rational in their decisions than most other decision-makers are. Their thinking is analytical and logical, their problem domain is relatively well understood, and their environment emphasizes rational thinking. Nonetheless, their individual behavior is boundedly rational at best while their aggregate behavior may be quite rational and predictable (see March 1994).

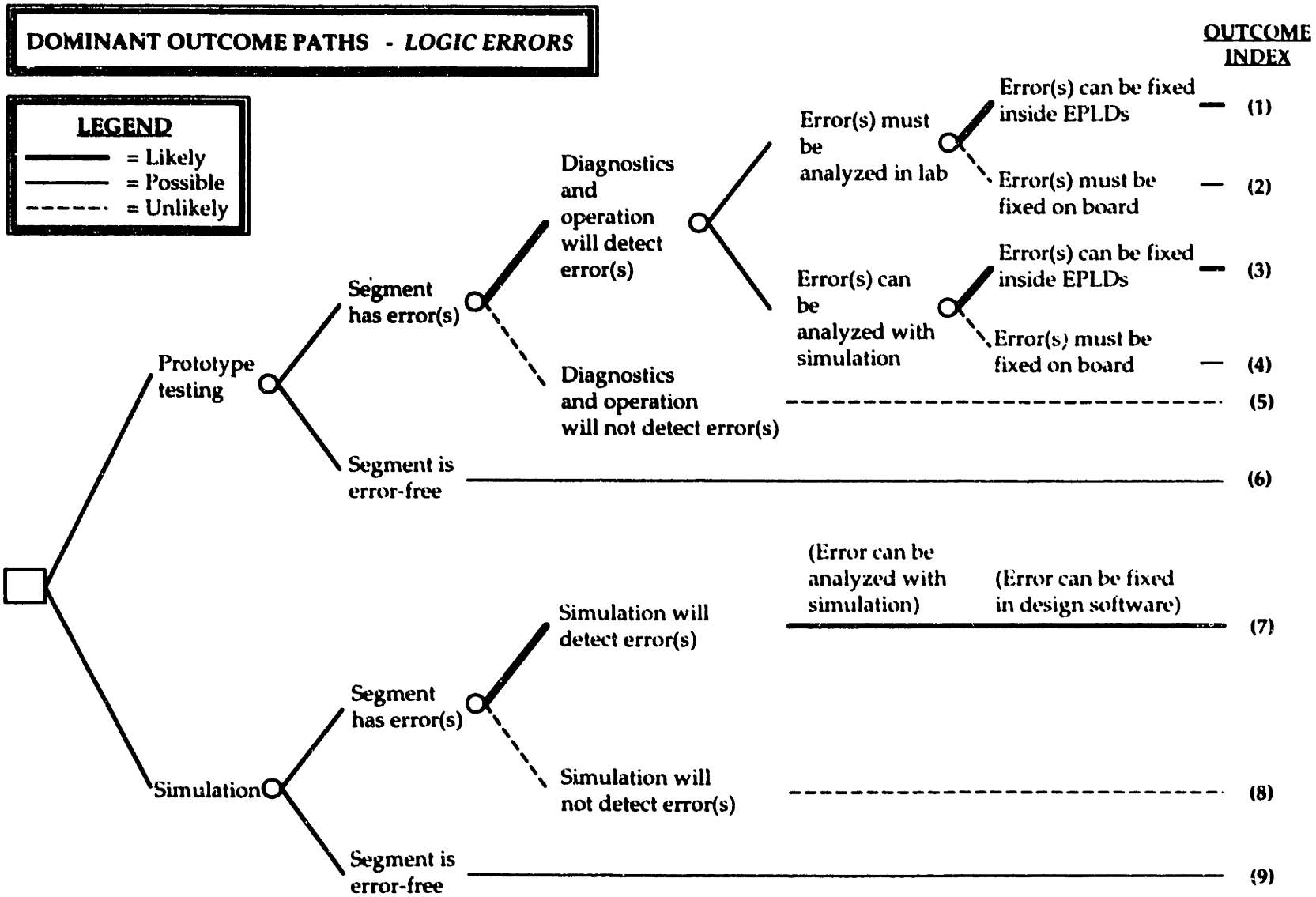


Figure V.7. Dominant outcome paths in the elimination of logic errors.

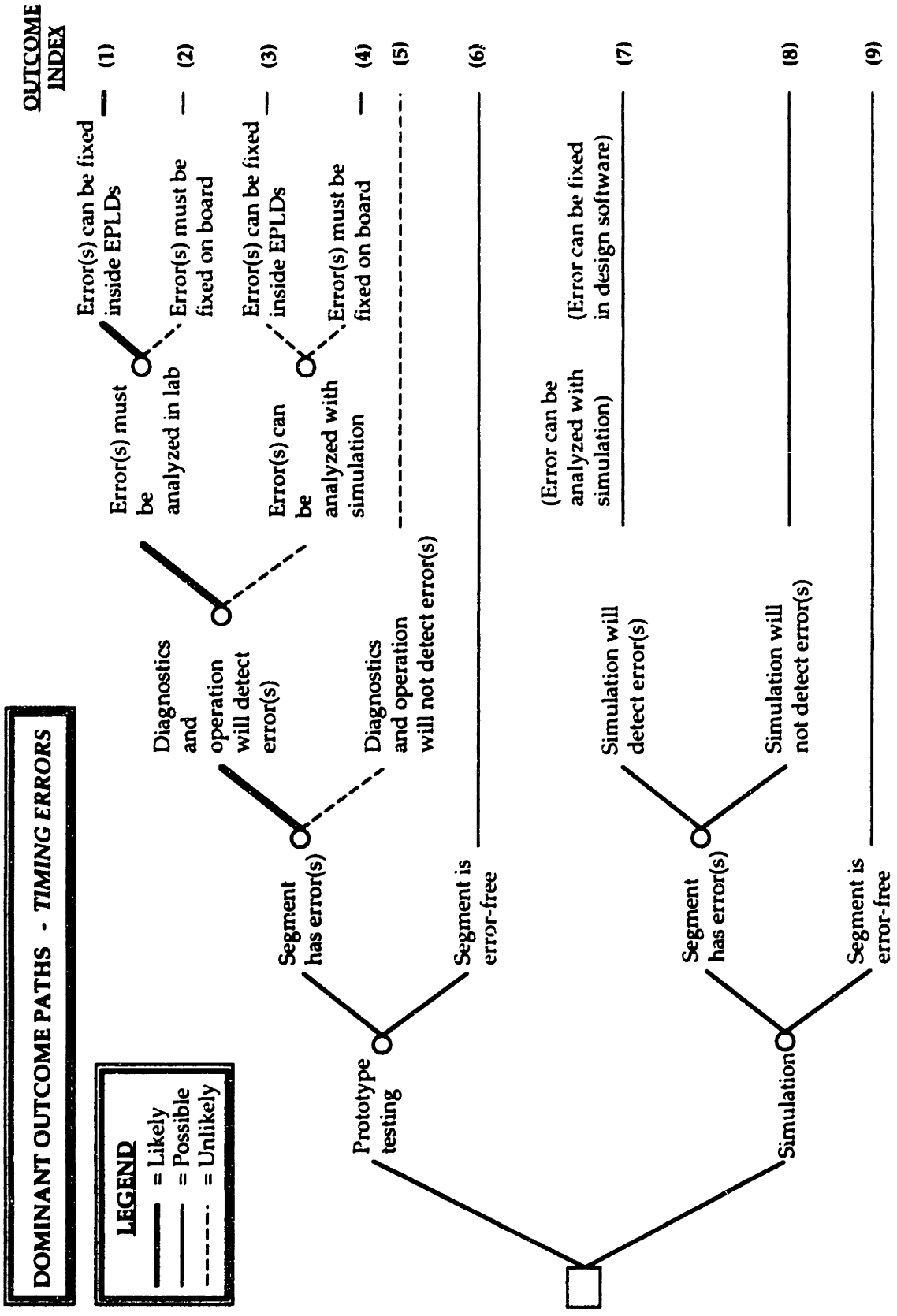


Figure V.8. Dominant outcome paths in the elimination of timing errors.

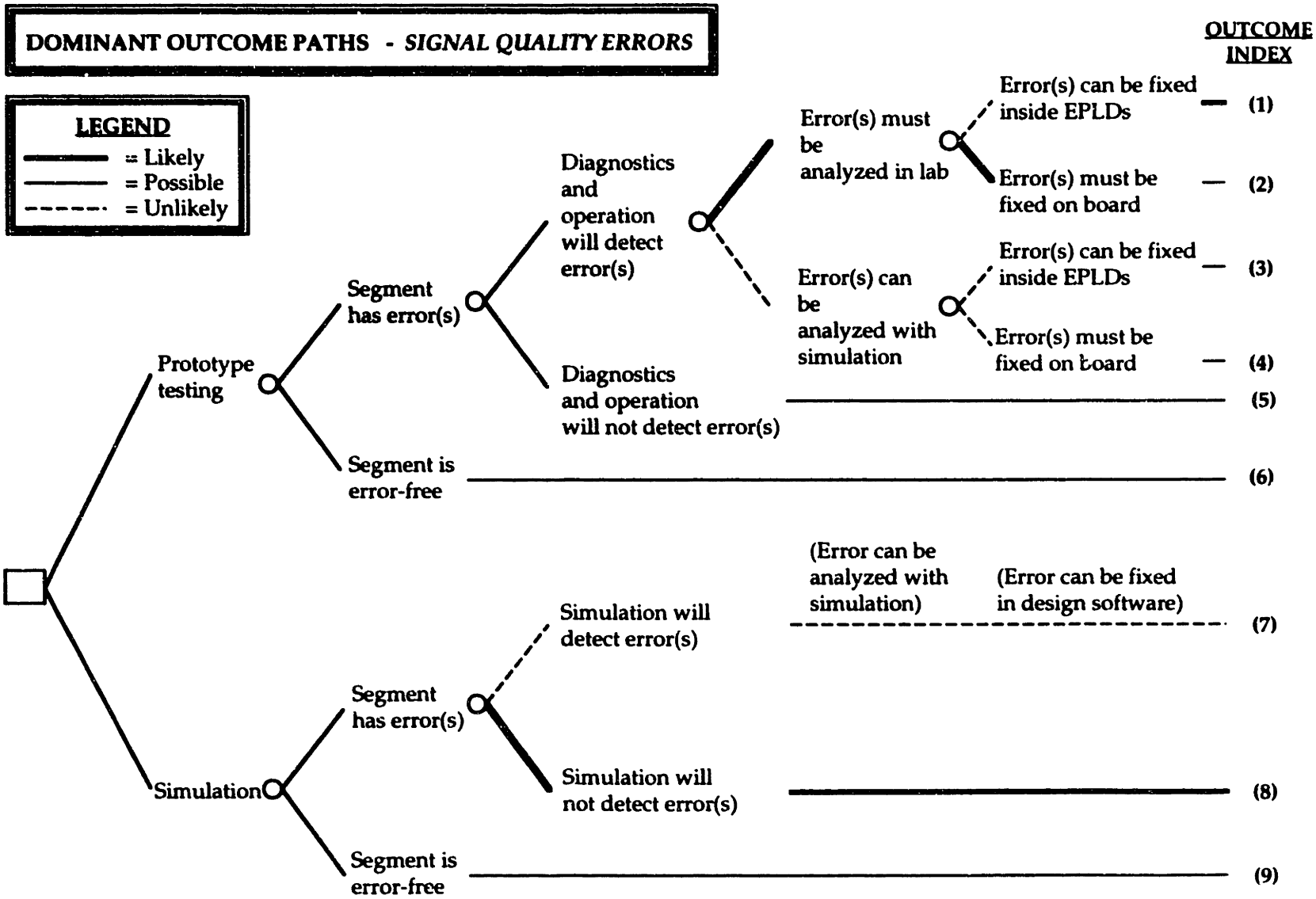


Figure V.9. Dominant outcome paths in the elimination of signal quality errors.

To remind us of the basic ideas behind mode switching, consider a mode switching problem with three experimentation modes A, B, and C. If a design started in mode A and an experimental cycle is completed, a designers is faced with the decision to either continue experimentation in mode A, switch to mode B or C and continue experimentation, or abandon experimentation because the expected incremental experimentation cost exceeds the expected incremental benefits. Key to efficient switching is therefore an accurate and exhaustive assessment of each mode's incremental cost and benefit, dynamically adjusted after an experimental cycle is completed.

I will now discuss a very simple example that illustrates some of the thinking a designer at my research site (BBN) undertook prior to selecting an experimentation mode and analyze the example with the aid of my general decision model. Even though the provided estimates were made after experimentation had been completed, the example nonetheless serves as a starting point in discussing the utility of decision models in the selection of experimentation strategies.

In the example, the interviewed designer had just completed the design of the data path of a bus exchange unit[12] (the "design segment") and had to decide whether to test the design with computer simulation (mode 1) or paper simulations (mode 2). The design of the data path was completed with a schematic design tool and it had taken the designer about two weeks.

[12] The designer recalled the design quite well since he had worked on it a few days prior to our interview. However, it was difficult for him to evaluate the different modes in hindsight because he had not tried to assess the costs and benefits of each mode with the level of detail that I propose.

According to the designer, the likelihood of a design error was very low since he had completed many similar designs before, a significant part of the design could be done by replicating logic cells, and there were few interactions between different parts of the design segment.

With respect to the four experimental steps, he had the following cost expectations prior to starting experimentation:

Step 2: Build

— *Computer simulation*: Since the EPLD was designed on a CAD tool, its computer simulation model could be directly created from the design file. In order to start computer simulation, however, the designer had to write 5-6 pages of behavioral code that would define the environment external to the design. (The behavioral code would become superfluous once the complete design became available.) The designer estimated that writing the behavioral code would take about 2 days, with an additional 0.75 days spent on initializing signals to be ready for simulation. Assuming an eight hour day, the total set-up time for computer simulation would therefore come to about 22 hours.

— *Paper simulation*: In the case of paper simulation, the design model remained inside the designer's "head" and on various notes he had taken while designing the segment. Relative to simulation, the set-up time was negligible.

Step 3: Run

— *Computer simulation:* The designer felt that running 10-15 [carefully selected] cases would give adequate test coverage because of the few

interactions between design segments. If he simulated 10-15 cases on a computer, the run time would be negligible. He claimed, however, that a visual screening of the computer simulation output and checking for errors would take him about 1.25 days (10 hours).

— *Paper simulation*: If simulated on paper, he can run and screen about 8 cases per day (1 per hour)[13]. Since he finally decided to run 13 cases, paper simulation was going to take about 13 hours.

Step 4: Analyze

— *Computer simulation:* The designer felt that the analysis of an error would be very fast if it was detected by computer simulation. The time to analyze was therefore considered to be negligible.

— *Paper simulation:* Similar to design, the analysis of an error would be fast if it was detected by paper simulation. Again, he considered the time to analyze an error as negligible.

Step 1: Correct Error

— *Computer simulation:* Since the design was on a CAD system, design changes were expected to be easy. Thus the designer considered the cost of design modifications to be negligible.

— *Paper simulation:* The cost of changes was identical to errors found by computer simulation.

---

[13] The designer was the most experienced in the interviewed group of four designers. Being able to paper-simulate eight well-documented cases per day is considered quite fast.

Finally, probability values had to be assigned to the branches of the decision model. A reasonable value for the probability of having at least one error in the design is $p_{a_m} = 0.1$. Furthermore, given the high skill-level of the designer, let us assign the probability of paper simulation detecting an error in this particular design (if 13 cases are run) a value of $r_{a_{paper}} = 0.7$ and the probability of computer simulation to detect an error a value of $r_{a_{computer}} = 0.9$. The quality loss of not eliminating a design error if it exists is denoted as L. The general decision model with inserted values is depicted in figure V.9. (next page) and the expected cost for each mode can now be calculated.

In chapter IV, I developed the following equation for evaluating the expected economic cost of an experimental cycle (equation IV.1).

$$E[Cost_{a_m}] = p_{a_m} \times r_{a_m} \times (c_i^{analyze} + c_i^{fix\&verify}) + p_{a_m} \times (1 - r_{a_m}) \times q_i + c_i^{build} + c_i^{run}$$

— Substituting the estimated values for paper simulation, we get:

$$E[Cost_{paper}] = 0.1 \times 0.7 \times (0 + 0) + 0.1 \times (1 - 0.7) \times L + 0 + 13$$
$$= 0.03 \times L + 13 \therefore$$

— Substituting the estimated values for computer simulation, we get:

$$E[Cost_{computer}] = 0.1 \times 0.9 \times (0 + 0) + 0.1 \times (1 - 0.9) \times L + 22 + 10$$
$$= 0.01 \times L + 32 \therefore$$

Since an error would eventually be detected in a design failure, a value of the quality loss L can be reasonably estimated by looking at the cost of finding the error either in the prototype or in the field. In this case, the hardware prototype would most certainly detect an error if it remained undetected in simulation.

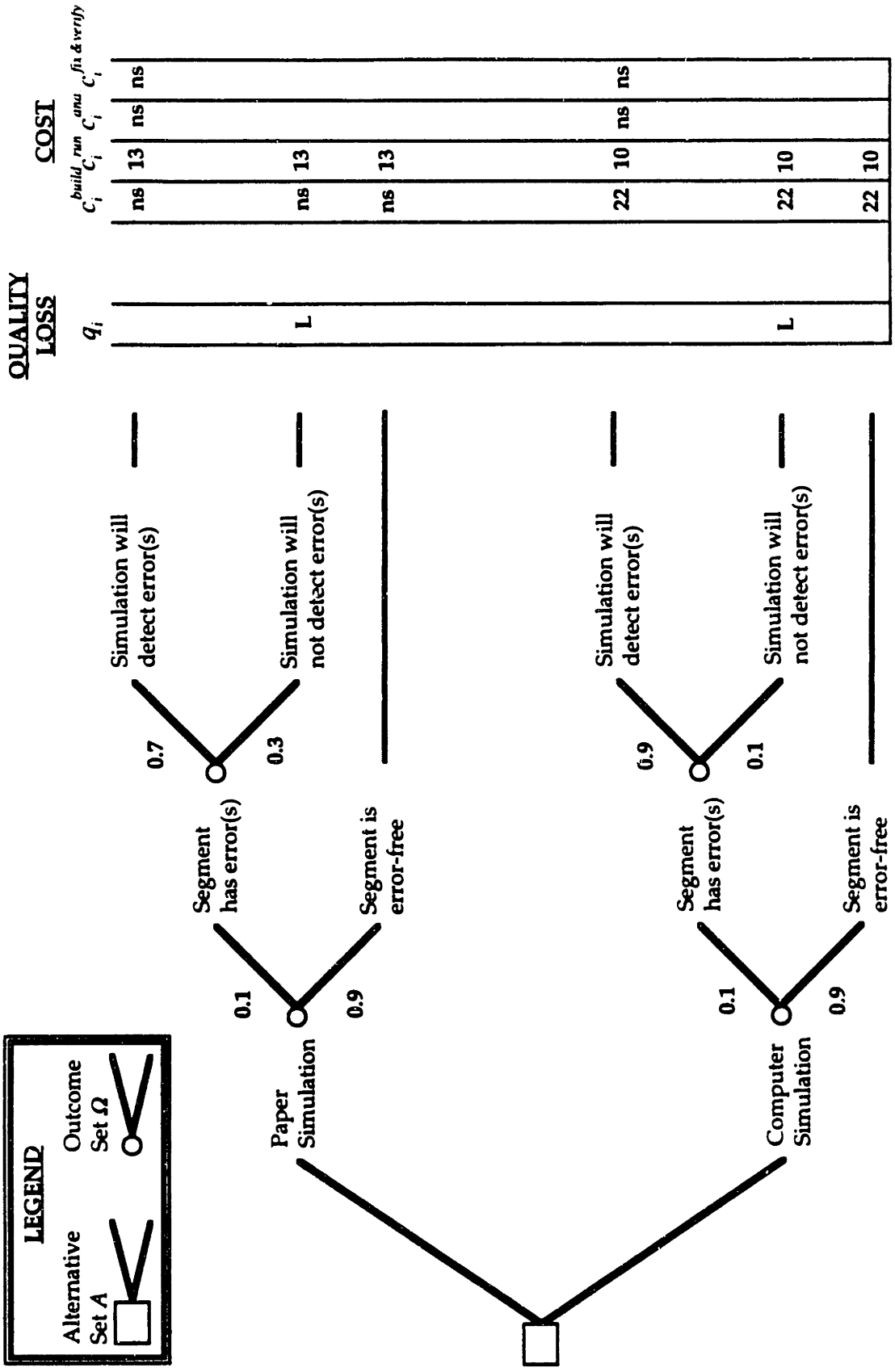


Figure V.9. Example: switching from paper simulation to computer simulation.

If we inspect the two equations above, one can easily see that paper simulation is a better choice for L < 950 man-hours (computer simulation is the better choice for L > 950 man-hours). The cost of eliminating an error in the hardware prototype was expected to be well below the cost of 950 man-hours and thus paper simulation was the more effective starting mode for this particular designer. Since mode selection is a dynamic process, the designer should have repeated the same calculations (with refined parameter estimates) each time a design error is eliminated (i.e. after each cycle).

Finally, I would like to emphasize that decision problems that involve prototype testing are probably more complex but, as we have seen in this simple example, the evaluation process is still simple enough to be operationalized in a design environment. A more sophisticated version of this model (e.g. one that would approximate error probability distributions and expected costs based on Bayesian learning) could be integrated in a commercial decision-support system.

V.6 Conclusion

In this chapter, I used the findings from an on-site field study to compare simulation and prototype testing as experimentation modes in the design of custom circuits. Detailed information learned from numerous face-to-face interviews and the results from a detailed study of 24 design errors (see table V.8) enabled me to extend and to refine the general decision model (chapter IV) in the context of designing custom circuits that contain electrically-programmable logic devices (EPLDs). The results also provided

some preliminary support for the general hypotheses structure that was developed in chapter III.

Number	Result Content — (as found in main text)
1	In the examination of design errors detected by prototype testing, it was found that a significant proportion (46%) could not have been detected by simulation .
2	Out of the errors that could not have been detected by simulation, a very significant proportion (73%) passed simulation because the simulation tools did not model the tested behavior with sufficient accuracy — a problem that affected only timing and signal quality errors.
3	Out of the errors that could have been detected by simulation, a very significant proportion (92%) passed simulation because an insufficient number of test cases was run. Presumably, designers stopped simulation "early" because, at some point, they felt that prototype testing was more effective.
4	Once an error is detected by prototype testing, it is very likely that the error cause will be analyzed in the laboratory (88% of examined cases). Sometimes, designers will try to recreate and analyze the error with simulation — in these findings, this occurred only in the case of logic errors (12% of examined cases).
5	Once the error cause is found, it is likely that the cost to fix the error in EPLD designs is low (75% of examined cases). The cost of fixing an error, however, increases with error class (logic error → timing error → signal quality error).

Table V.8. Summary of results 1-5.

Finally, I used empirical data to demonstrate how designers can use the decision model to evaluate experimentation modes.

Chapter VI

Second Field Study: Survey on Experimentation in Custom Circuit Design

VI.1 Introduction

In this chapter, I will report on the results of a large scale survey study conducted in the custom circuit design industry. Two groups, designers of

circuits containing electrically-programmable logic devices (EPLD designers) and designers of application- specific integrated circuits (ASIC designers), each received 500 questionnaires in which they were asked to respond to questions about their experimentation strategies in design. Out of the returned questionnaires (47.05%), a total of 391 questionnaires could be used for testing the hypotheses structure proposed in chapter III. The survey respondents were found to have significant design and simulation experience, and no indication of a possible response bias was found.

Some of the findings are briefly summarized below (please consult the main text for more details, especially for a discussion on how the hypotheses structure relates to these findings).

— Findings Related to Information Available *Prior* to Starting Experimentation:

(1) Overall, ASIC designers rated the effectiveness of simulation relative to prototype testing significantly higher than EPLD designers. This result is consistent with my earlier proposition that, because of the high cost of building/modifying prototypes, ASIC designers find simulation more beneficial.

(2) EPLD and ASIC designers consistently rate simulation to be most effective in experimental step (4) (analysis), followed by step (1) (fix error) and step (3) (run).

(3) EPLD and ASIC designers rate the effectiveness of simulation relative to prototype testing to be decreasing with error class (logic errors → timing errors → signal quality errors). If one uses error

class as a proxy for model accuracy (as suggested in chapter V), then the result agrees with my earlier hypothesis (3).

(4) At the start of experimentation, EPLD designers are less likely to use simulation than ASIC designers are.

(5) For both, EPLD and ASIC designers, the likelihood of initially using simulation decreases with error class (logic errors → timing errors → signal quality errors).

(6) Both, EPLD and ASIC designers, have little knowledge about mode efficiency trajectories prior to starting experimentation

— Findings Related to Information Available *After* Starting Experimentation:

(1) After starting experimentation, EPLD designers tend to switch from simulation to prototype testing earlier than ASIC designers do.

(2) For both, EPLD and ASIC designers, the likelihood of switching from simulation to prototype testing "early" increases with error class (logic errors → timing errors → signal quality errors).

(3) In their decision to switch from simulation to prototype testing, EPLD and ASIC designers rely heavily on a "declining error detection rate" in their decision-making process. They are unlikely to plan the switching point prior to starting experimentation.

(4) In comparing design projects of similar complexity, EPLD design projects required on average 8.45 man-months whereas ASIC design projects required an average 19.24 man-months (difference is significant at $p<0.01$).

(5) In the same comparison, EPLD designers averaged 13.90 prototype iterations whereas ASIC designers averaged 1.49 prototype iterations (difference is significant at $p<0.01$). Thus, EPLD designers used prototype testing as an experimentation mode while ASIC designers tried to get their design "right in the first prototype".

Overall, the findings were mostly in accordance with the models and the hypotheses proposed in earlier parts of this thesis. The last finding is particularly noteworthy: while the difference in design effort between EPLD and ASIC designers was in line with my general thesis, the large magnitude of the difference was somewhat surprising and merits further investigation.

VI.2 Hypotheses Testing

In this second field study, I prepared a large-scale survey to compare simulation and prototype testing in custom circuit design, and use the findings to either support or fail to support my hypotheses structure (see figure VI.1 on the next page) in the context of custom circuit design.

In the survey, I collected two kinds of experimentation-related information: (1) information that is available prior to starting experimentation (and is used to select a starting experimentation mode); and (2) information that becomes available after experimentation has started (and is used to iteratively evaluate mode efficiencies). I will now discuss both kinds of information and how my hypotheses relate to them in turn.

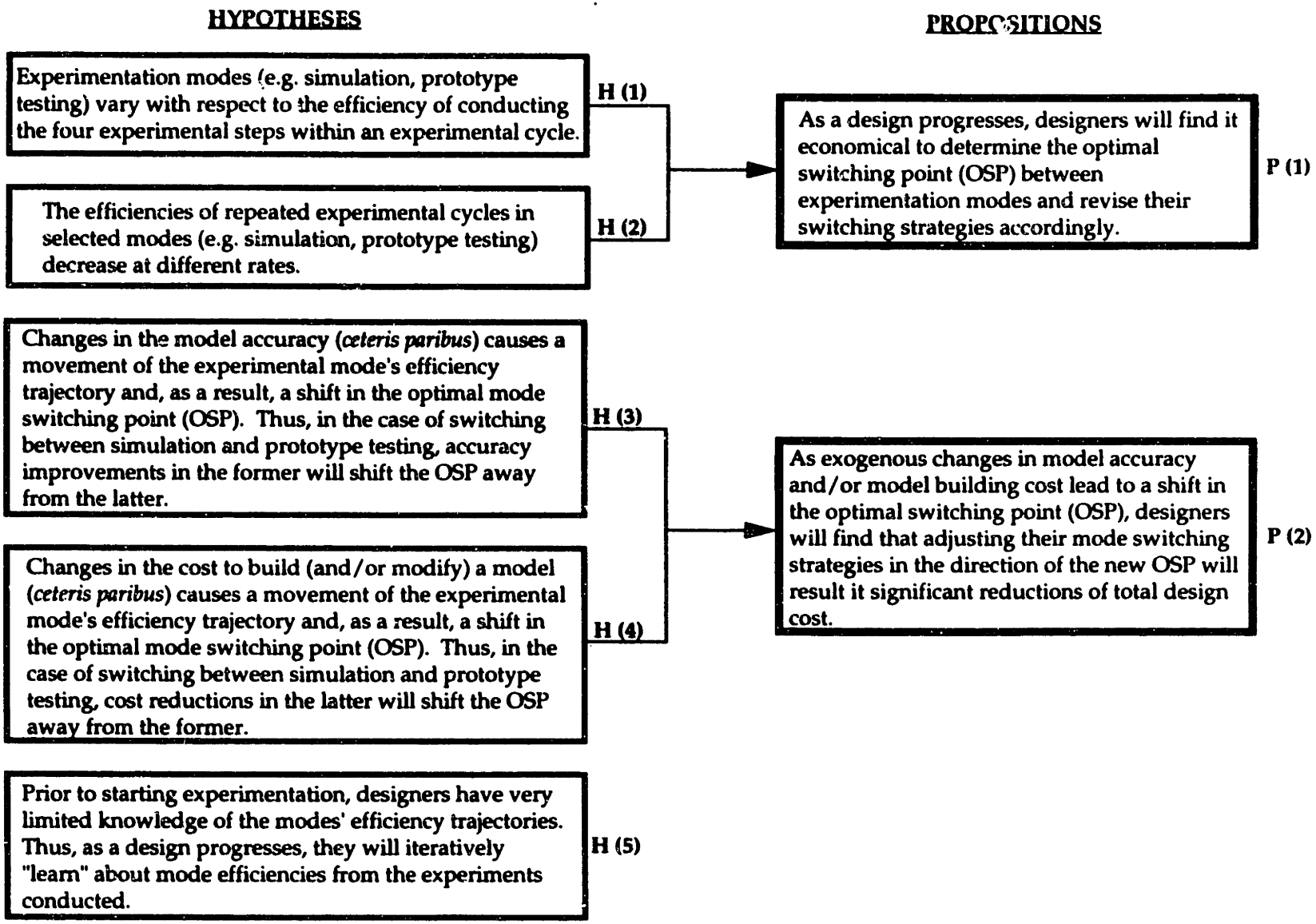


Figure VI.1. Hypotheses structure to be tested.

Information Available *Prior* to Starting Experimentation (*A Priori* Information)

Designers have some information about the efficiency of modes prior to starting experimentation. This information is a result of numerous experiments conducted in past designs and it does not depend on the dynamics of the environment in which experiments are to be conducted. For example, designers know something about the relative efficiency of simulation over prototype testing in eliminating errors prior to simulating a particular design but they know very little about the relative efficiency change (and the related trajectory) after the start of experimentation unless they have posterior information from conducting experimental cycles. Thus, *a priori* information aids designers in selecting the most efficient starting mode (for example, consider the illustration in figure VI.2 where simulation is initially more efficient than prototype testing).

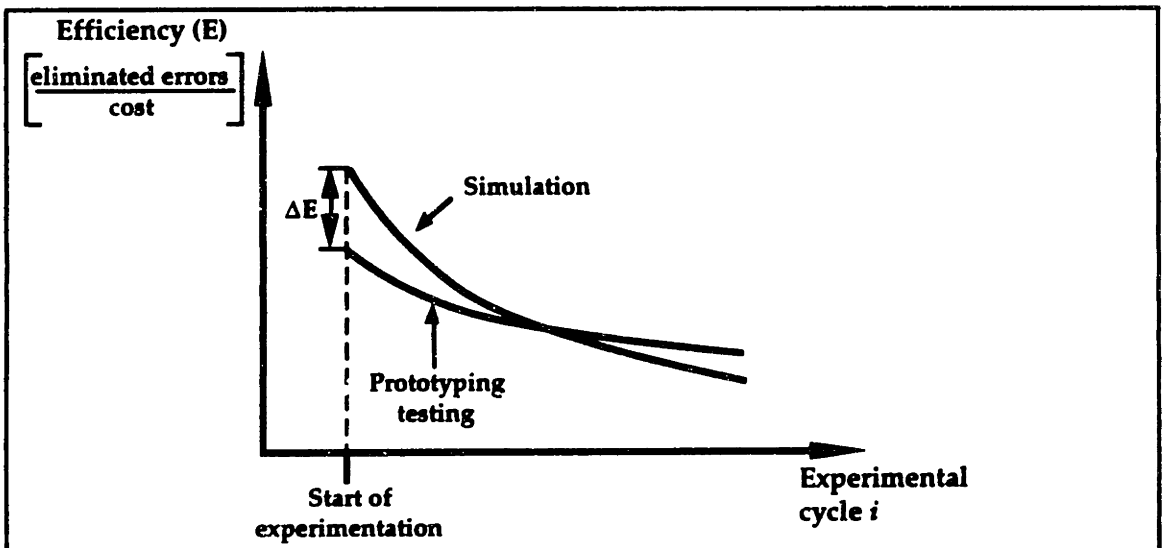


Figure VI.2. The selection of an experimentation mode prior to starting simulation (illustration).

Thus, by examining the information that designers have prior to starting experimentation, I will be able to test hypotheses (or parts of hypotheses) that do not depend on conducting experimental cycles.

Information Available *After* Starting Experimentation (*Posterior* Information)

As designers engage in repeated experimentation, they learn new information about changes in the efficiency of the experimentation mode(s) being used, and can refine the efficiency estimates they made prior to starting experimentation. These new estimates can then be used to dynamically adjust their mode switching strategies (e.g. the decision to switch from computer simulation to prototype testing). For example, a declining logic error detection rate may be used as an indicator for a declining efficiency of simulation relative to prototype testing (see figure VI.3 on the next page for an illustration). And when the efficiency of simulation falls below the expected efficiency of prototype testing, a user may find it economical to switch modes.

Thus, by examining the designers' mode switching behavior after experimentation has begun, I will be able to test the hypotheses (or parts of the hypotheses) that do depend on conducting experimental cycles.

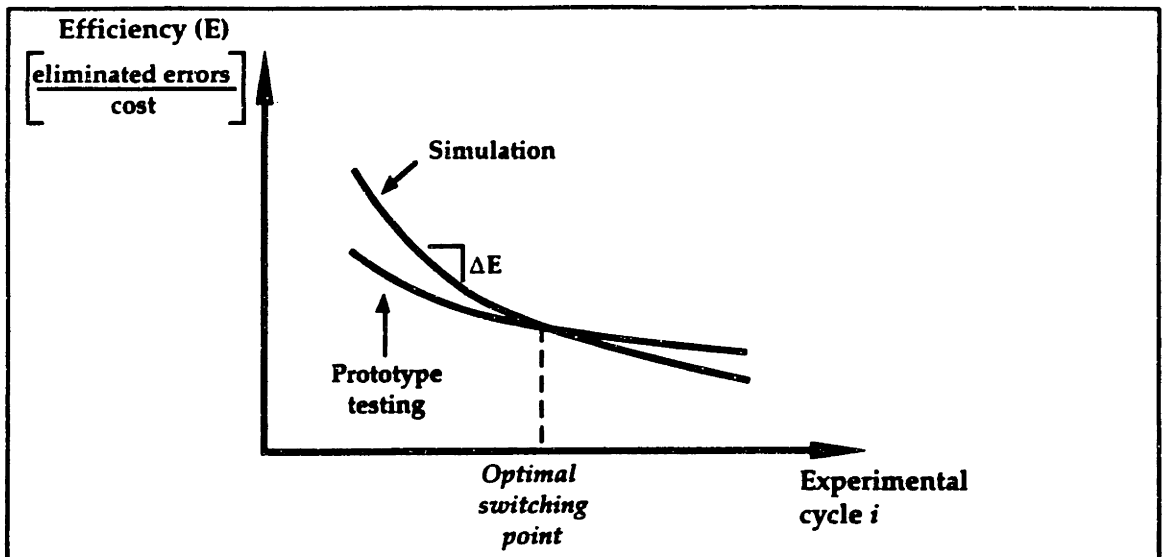


Figure VI.3. The iterative re-evaluation of a mode's efficiency (illustration).

VI.3 Research Methods

VI.3.1 Sample Context and Research Design

In my second field study I continued to compare computer simulation and prototype testing in custom circuit design. In order to apply rigorous testing to my hypotheses, I mailed 500 questionnaires to each: (1) designers of custom circuits containing electrically-programmable logic devices (EPLDs); and (2) designers of custom circuits containing application-specific integrated circuits (ASICs) (please consult appendix B for more information on these design technologies).

In the survey, I examined two levels of variation with respect to the efficiency of experimental modes: (1) efficiency variations within a group of designers who face similar costs to build/modify prototypes; and (2) efficiency variations between two groups of designers who perform similar designs but

face very different costs to build/modify prototypes (which would allow me to test hypothesis 4).

With respect to (1), my first field study showed that within group efficiency variations can be studied by closely examining efficiency as a function of error classes (which serve as a proxy for experimentation model accuracy) and experimental steps.

With respect to (2), I had learned from earlier research that between group variation can studied by comparing EPLD and ASIC designs since they differed significantly with respect to the cost to build and to modify a hardware prototype[1] (see appendix B). As ASIC designers have more to gain from simulation relative to EPLD designers (i.e. by not having to make costly modifications to the prototype during experimentation), I expected to find a significant difference in their mode switching behavior.

Thus I decided to follow a dual sampling strategy: select two groups of designers (EPLD and ASIC) and survey them with the aid of two questionnaires (one for each group) that are identical in research content. The results would allow me to analyze within group (with respect to error classes and experimental steps) and between group (with respect to cost of prototyping) variations in experimentation mode efficiencies in custom circuit design (shown in figure VI.4 on the next page).

---

[1] The cost to build (and to modify) a prototype is commonly referred to as non-recurring engineering (NRE) costs. As a general rule, NRE costs in ASICs are high, while they are low for EPLD-based designs.

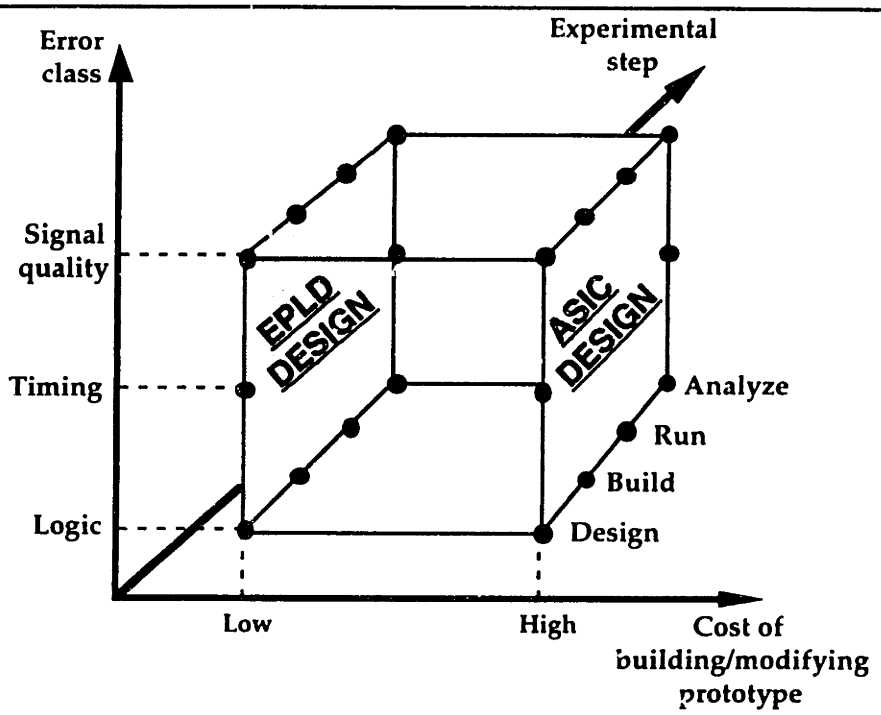


Figure VI.4. Research design of circuit design survey.

VI.3.2 Description of Sample

Survey Participants

The survey participants were selected from the subscriber database of the largest industry journal (ASIC & EDA) focusing on issues related to ASIC and EPLD-based design. The complete database encompassed names and profiles of over 10,000 circuit designers world-wide. Based on previous field research, I developed participant profiles (see table VI.1 on the next page) for the types of designers I wanted to be part of the survey sample and generated mailing lists that contained 500 designers from each group (EPLD and ASIC)[2].

[2] A sample size of 500 designers for each group (1000 total) covered over 90% of all names in the database that matched the participants' profiles.

Characteristic	*EPLD Designers*	*ASIC Designers*
Design Technology	- PLDs - FPGAs	- Gate array - Cell based - Full custom
Principal Job Function	- System designer - Design and development engineering	
End Product	- Personal computers - Workstations - Other computers (mini, mainframe, super) - Computer peripherals - Communication systems - Avionics, marine, space, electronics - Military - Industrial controls, systems, equipment robotics - Office and business machines - Computer systems integrator - Automotive and other ground vehicles - Medical electronic equipment - Electronic instrumentation - Consumer electronics and appliances - Design service provider	
Design Responsibility	- Chip level - Board level - Systems level	
Principal Responsibility	- Engineering - Engineering management	
Company Size & Location	- Any size, any location within the United States	

Table VI.1. Profiles of selected participants.

Questionnaire Design

Earlier we have seen that the hypotheses-related information can be divided into two categories: (1) information available prior to starting experimentation; and (2) information available after starting experimentation. The questionnaire design followed along these lines[3].

The *first* section ("Background and Experience") includes general questions about the respondents' experiences and design styles which can be

[3] Copies of the questionnaires can be found in appendix D.

used to establish whether respondents have sufficient expertise to answer the questionnaire.

The *second* section ("Effectiveness of Simulation and Prototyping in Debugging Errors") asks designers to evaluate the general effectiveness of simulation and prototype testing as a function of error classes and experimental steps prior to starting experimentation (*a priori* information). With the responses to these questions, one can test hypotheses (or parts of hypotheses) that do not depend on information from experimental cycles (which relates to hypotheses 1, 3, 4 and 5).

The *third* section ("Switching from Computer Simulation to the First Prototype") is the main part of the questionnaire and includes questions related to information that is available prior and after the start of experimentation. The questions are aimed at the last EPLD or ASIC design that the respondent was directly responsible for or involved in. Using a "recall the last design" method has several advantages. First, design projects tend to last over several months and information about the last design is often the easiest to recall. Second, design technology and tools change quite rapidly in IC design and therefore allowing designers to refer to very old designs *might* introduce unwanted variability. Third, by asking designers to recall a specific design ("the last one"), they weren't tempted to bias their responses by selecting a particularly successful design.

Some of the issues that the third section addresses are:

— General characteristics of the last design such as size, number of parts, rated logic complexity, etc. (this information will be important later when I compare design projects and their related performance).

— The designer's knowledge about the distribution of design errors prior to starting simulation or prototype testing. (Important in understanding the process of "learning about mode efficiencies" (relates to hypothesis 5)).
— The extent to which simulation was used and how decisions to switch from simulation to the first prototype were made. (Important in understanding and testing hypotheses on declining mode efficiency rates and the related switching strategies (relates to hypotheses 3, 4 , and 5)).

Finally, the *fourth* section includes questions about prototype iterations and the related impact on total design cost (responses to these questions will help me in testing proposition 2).

I prepared two versions of the questionnaire: version *A* for EPLD designers and version *B* for ASIC designers. The two versions were essentially identical in content but I had to adapt the technical language to the particular design environment being sampled. Before proceeding to pretest, I had several experts critique the questionnaires and made several modifications along the way. After several weeks of rewriting and numerous iterations, the pretest versions were finally available.

VI.3.3 Data Collection

Pretest

Pretesting the questionnaire is an absolutely essential step in questionnaire research and serves a number of purposes (Judd, Smith, and Kidder 1991):

— It helps in identifying unforeseen problems in question wording or respondents' comprehension, question sequence, or questionnaire administration so that they can be eliminated before the actual study.
— It may indicate the need for additional questions or the elimination of others.
— It can help in gauging whether the length of the questionnaire affects the likely response rate.
— It allows the collection of open ended responses.

I decided that the pretest had to be conducted at a site different from my first field study (BBN in Cambridge) because (1) designers at BBN were already familiar with my research which could have introduced bias in their responses; and (2) my field research at BBN influenced the questionnaire design and therefore I needed feedback from other design environments. I contacted a design group at Hughes Network Systems (HNS) in Germantown, Maryland and asked for their cooperation in the pretest phase. HNS, a unit of GM Hughes Electronics, is a global leader in satellite, digital cellular and networking technologies. The firm designs, manufactures, and installs advanced networking solutions for businesses and governments worldwide, and is credited with many of the multi-protocol network technologies in use today. HNS was particularly interesting as a pretest site since its networking technology design groups consist of both, ASIC designers and EPLD designers.

After securing the firm's cooperation, survey response kits were prepared and mailed to 10 designers in the satellite networks division (EPLD designers were in the hardware engineering group; ASIC designers were in the ASIC development group). The designers were asked to fill out the

questionnaire as if it had been received through random mailing and then write a critique immediately after completing the questionnaire[4]. Furthermore, all participants were asked to sign up for a 30 minute personal interview to be conducted as part of a test site visit 4 days after the designers received the response kit. One day prior to the interview date, I reviewed copies of completed questionnaires in order to get a set of issues likely to be raised during the interviews. On the day of the interviews, 9 designers (the 10th designer had a schedule conflict) were first questioned about general issues (e.g. time to complete, general impression, likelihood of response) and then they were asked to explain their responses to each question in the questionnaire. Their explanations allowed me to examine if all questions had been understood as intended and if their responses would permit a meaningful analysis. All interviews lasted 30 to 40 minutes. Detailed notes were taken and evaluated later during the questionnaire redesign phase. The designers made a number of suggestions mostly related to the ambiguous wording of some questions or the difficulty of answering others. Some designers expressed concern about the length of the questionnaire. As a result I either rewrote or eliminated several questions and trimmed the questionnaires from 5 to 4 pages[5].

---

[4] They were supplied with a list of issues that they should respond to but were free to raise any additional issues they were concerned about.

[5] For example, I asked designers to assess their own skill level. However, in my analysis I found that junior designers tend to overestimate their skills while senior designers ("experts") have a better understanding of their weaknesses and thus underestimate their skills relative to others. Since the question was not important to my research design, I decided to eliminate it.

Mailing and Response

On August 12, 1994, 1000 survey response kits were sent to the selected sample of EPLD and ASIC designers. 500 EPLD designers received response kits containing version *A* of the questionnaire and 500 ASIC designers received version *B*. Each response kit contained a cover letter, a questionnaire and a self-addressed stamped envelope[6]. All questionnaires were numbered in order to keep track of the responding groups and, in case of a low response rate, to have the option of sending another set of questionnaires to the non-responding group.

In the cover letter I explained the salience of my research to the respondents[7] and emphasized that their individual responses are confidential. Since I had reason to believe that some questionnaires could end up with designers that had unsuitable backgrounds or design responsibilities, I included a description of the background or responsibility necessary for answering the questions. Designers who felt that they didn't meet the provided description were encouraged to mark it with "wrong experience" and send it back to me. Follow-up letters that thanked the early respondents and encouraged the others to reply were mailed 7 days after the response kits were sent out .

A detailed breakdown of the responses is shown in table VI.2. 16 response kits were returned to me as "undeliverable" (with the unusually

---

[6] Research has shown that the class of postage sometimes has an effect on response rate. For example, some studies comparing first-class postage, business reply postage, and metered postage found that first-class postage yielded slightly [statistically] higher response rates (Lockhart 1985). Thus I decided to use first-class stamps.

[7] Research has shown salience to have a significant effect on response rates (Lockhart 1985).

high turnover in the high-tech industry, most of these addressees had probably left their companies). Out of the 463 responses mailed back to me (47.05%), 61 (6.20%) were marked "wrong experience" and 11 (1.12%) arrived after the data analysis had been completed. Thus I was left with a total of 391 (39.74%) questionnaires that were included in the analysis.

Description	*EPLD*	*ASIC*	*Both*
Total number of questionnaires mailed:	500	500	**1000**
– returned because addressee unknown:	– 4	– 12	**– 16**
Adjusted total:	496 (100%)	488 (100%)	**984 (100%)**
Number of responses received:	232 (46.77%)	231 (47.34%)	**463 (47.05%)**
– marked as "wrong experience":	– 22 (4.44%)	– 39 (7.99%)	**– 61 (6.20%)**
– arrived after analysis was completed[8]:	– 7 (1.41%)	– 4 (0.82%)	**– 11 (1.12%)**
Number of responses useful for analysis[9]:	203 (40.93%)	188 (38.53%)	**391 (39.74%)**

Table VI.2. Breakdown of response statistics.

Next I looked at the possibility of different response rates of the two groups of designers (EPLD and ASIC) which would raise the possibility of a group bias. The actual response rates (see table VI.2) did not indicate a significant difference between the groups. Respondents from both groups

[8] The data analysis was started approximately 6 weeks after the questionnaires had been mailed and was completed 2 weeks later. All responses received after that date were not included in the data analysis. However, all late responses were inspected for unusual patterns that were inconsistent with the analysis' results. No inconsistencies were found.

[9] The reader will find that many analyses use fewer datapoints than the total number of completed questionnaires. This was due to (ordered by relative frequency): (1) the respondent was instructed to skip the question (see questionnaire in appendix D); (2) no answer was provided for the question analyzed (e.g. because of a company's non-disclosure policy); (3) only respondents who answered all questions within a logical group of questions were included in the analysis; or (4) the answer was not consistent with provided instructions.

commented in their questionnaires that my research topic is of great importance to them and they hoped for some type of feedback about the outcome of the research project (e.g. a summary of the results in an industry journal).

Another possible concern in the use of questionnaires is a low response rate and the resulting possibility of a non-response bias. The response rate of 47.05% is comparable to surveys conducted by researchers in social science using a similar mailing process (Miller 1977). In comparison with surveys conducted in the IC design industry, however, my response rate can be considered quite high. An informal comparison with professional surveyors working in the same industry suggested typical response rates of 10-15% (with much shorter questionnaires and financial incentives to respond). Thus my survey's [higher] response rate can be interpreted as additional evidence for the relevance of my study.

Finally, I had no reason to believe that a non-response bias was present. First, my questions related to the use of simulation and prototype testing in design and there was no evidence that variation in use had an effect on questionnaire responsiveness. Second, extensive interviews during the pretest phase indicated that designers who do not respond are either busy with a critical phase of their design or are fed up with random questionnaire mailings and thus do not respond to surveys in general (irrespective of content)[10].

---

[10] A participant in the pretest confessed that designers receive 4-5 questionnaires per month. As a result, most designers do not respond unless they think that a study is salient to their work.

VI.4 Findings

The following section provides a summary and analysis of the questionnaire results. In my discussion, I will frequently reference the original questions as they are stated in both questionnaires. For the sake of brevity, however, I will only restate a question if it is directly relevant to the discussion. The exact wording of all questions can be found in appendix D.

VI.4.1 Characterization of the Response Groups

Section I of both questionnaires was designed to collect information on the respondents' experience in circuit design and simulation. Since the research questions required an advanced understanding of the design process in general and error elimination in particular, I needed to reassure myself that the responding group had sufficient design experience. Furthermore, as some of the analyses involve comparisons between the responses of EPLD and ASIC designers, I wanted to be aware of any differences between the two groups.

The data shows that EPLD designers have an average experience of 10.67 years; a large number if one considers the novelty of modern EPLD design technologies and design tools (see figure VI.5 on the next page)[11]. In my view, a [mean] decade of design experience is more than sufficient to respond to my survey. Research into expert performance and learning has found that experts require approximately 50,000 chunks of stimuli stored in their memory which takes roughly a decade to acquire (Simon 1981, Larkin,

---

[11] Presumably a significant number of designer has worked with predecessors of modern EPLDs (e.g. low-density programmable logic arrays, or PALs).

McDermott, Simon and Simon 1980). (A chunk is any stimulus that has become familiar from previous repeated exposure and hence is recognizable as a single unit.) Thus for all practical purposes, the group of EPLD designers can be considered as very experienced.

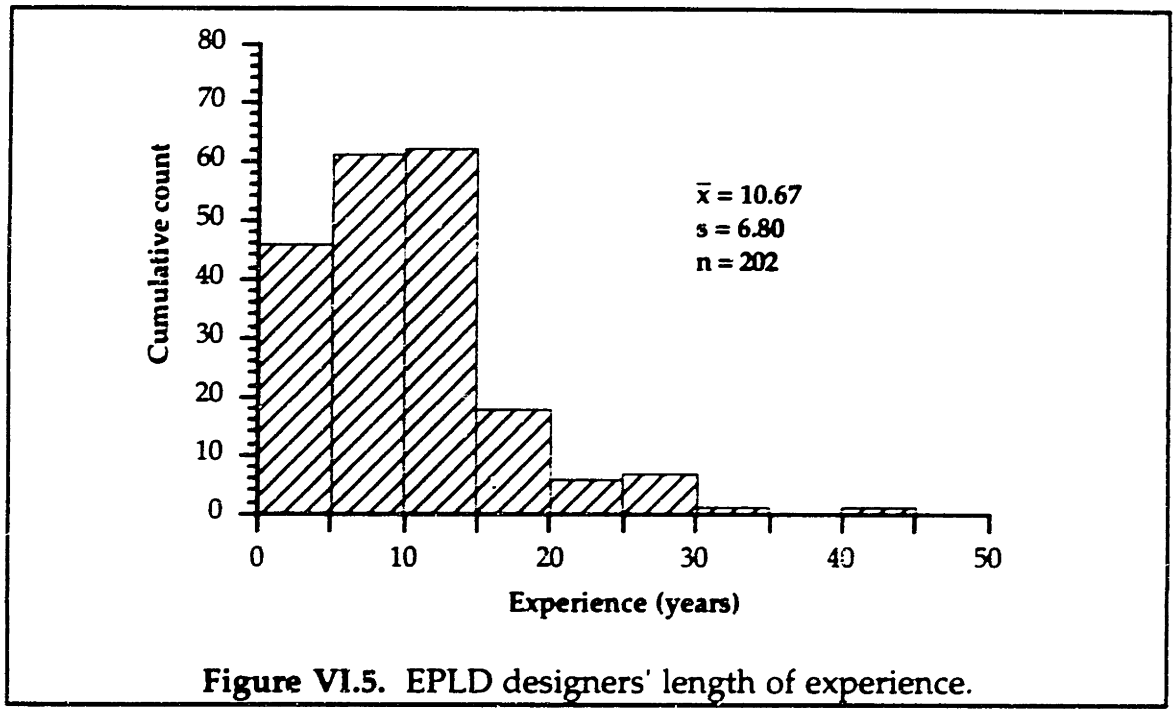


Figure VI.5. EPLD designers' length of experience.

An examination of ASIC designers shows a similar distribution of experience (see figure VI.6 on the next page). On the average, they have designed ASICs for 10.10 years which can be considered as very experienced using the same reasoning as before.

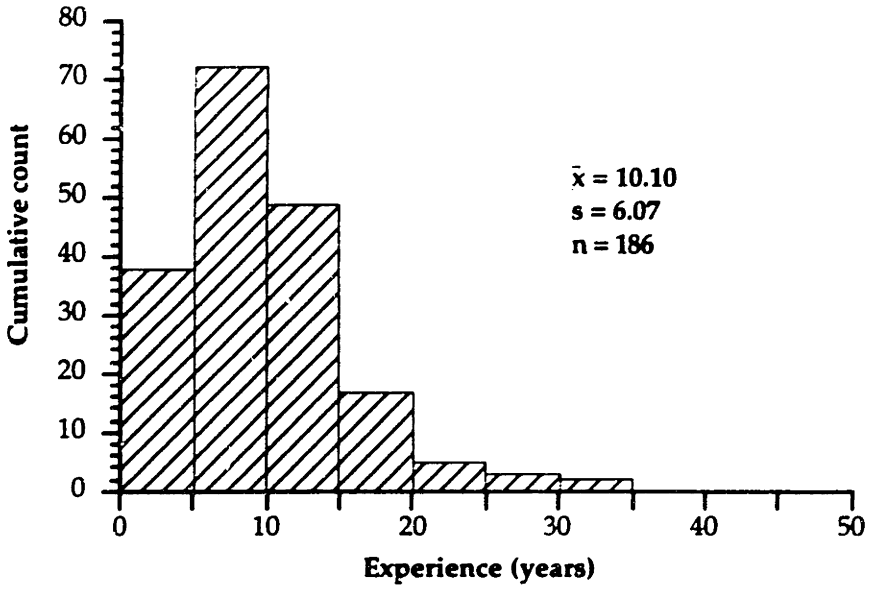


Figure VI.6. ASIC designers' length of experience.

As both groups possess a high degree of expertise, there is still the possibility of a difference in experiences and if so, the possibility of an experience bias. A comparison of the mean experiences results in no [statistically] significant difference[12] in experience between EPLD and ASIC

---

[12] Since this is the first statistical test used, I would like to explain my general test methods. The following tests will be used (Hogg and Ledolter 1987, Montgomery 1991, Sachs 1984, Ostle and Malon 1988)):

— *Continuous variables*: For comparison of means within the same sample, I use a two-tailed paired difference t-test. For comparison of means between two independent samples, I use a two-tailed t-test with independent estimates of sample variances.

— *Likert scale (1-7) variables*: For Likert scale variables, I use (1) parametric tests; and (2) non-parametric tests (npar). For comparison of means within the same sample, I use (1) a two-tailed paired difference t-test and (2) a Wilcoxon matched pair signed rank test. For comparison of means between two independent samples, I use (1) a two-tailed t-test with independent estimates of sample variances; and (2) a Mann-Witney-U test. The reason for using parametric and non-parametric test has to do with the assumptions underlying the respective tests. For example, t-tests assume a continuous distribution while the non-parametric U-test doesn't. On the other hand, non-parametric tests compare if the two probability distributions are equivalent which is not exactly the same as a comparison of means. (It is approximately the same if means and medians are equal such as in symmetrical distributions.) Because my test samples are fairly large, I do not expect significant differences between parametric and non-parametric tests.

designers and thus no evidence in support of a potential experience bias is found (see table VI.3).

Criteria	EPLD			ASIC			p-value
	Mean	St.dev.	n	Mean	St.dev.	n	(t-test)
Length of design experience (years)	10.67	6.80	202	10.10	6.07	186	0.382

Table VI.3. Comparing respondents' design experience.

Next I asked the respondents to state the length of experience in using design simulation tools. Because reasonably priced simulation tools have become available only in the last few years (particularly in EPLD design), I expected simulation experience to be significantly less than total design experience. The data is consistent with my expectations: EPLD designers have used design simulation tools for an average of 5.02 years whereas ASIC designers have used them for an average of 7.74 years. The difference in experience between the two groups is statistically significant ($p<0.001$) (see table VI.4).

Criteria	EPLD			ASIC			p-value
	Mean	St.dev.	n	Mean	St.dev.	n	(t-test)
Length of experience with simulation tools (years)	5.02	3.72	201	7.74	4.21	188	0.000**

Comment: — P-value: ** = $p<0.01$; * = $p<0.025$.

Table VI.4. Comparing respondents' simulation experience.

One possible explanation for the difference between the two groups lies with the historical evolution of design simulation tools in their respective fields. The availability and diffusion of EPLD simulation tools has lagged ASIC simulation tools because of the differences in benefits that each group derives from simulation (because of high prototyping cost, ASIC designers

will find it more beneficial to use computer simulation — an issue that will be addressed in other parts of this section). As a consequence, ASIC simulation tools are not only more advanced with respect to accuracy but they're also used on higher performing hardware platforms. The level of experience in using simulation in regard to answering the survey questions, however, is more than sufficient for both groups.

Finally I asked designers to evaluate their own design styles. While the question was intended to be explorative, there was a possibility that "design style" had an influence on some responses. My field research had shown that EPLD designers rely heavily on incremental trial and error (i.e. try a little, test a little) strategies throughout their designs. In contrast, ASIC designers were more reluctant to admit that they designed incrementally. They often claimed that getting "the prototype right the first time" was their objective in using simulation. However, a comparison of EPLD and ASIC designers' responses shows that both see themselves as incremental designers. Surprisingly, ASIC designers see themselves as designing [significantly] more incremental than EPLD designers (see table VI.5 on the next page). A plausible explanation for the difference lies with defining "incremental": ASIC designers use incremental approaches during computer simulation but clearly not in the prototyping phase[13]. In contrast, EPLD designers may see themselves as incremental designers during simulation and prototype testing.

---

[13] If they did, it would be reflected in their prototyping behavior. As we will see at the end of this section, ASIC prototypes are usually not used as experimentation modes due to the high cost of prototype changes.

Criteria	EPLD Mean	EPLD St.dev.	EPLD n	ASIC Mean	ASIC St.dev.	ASIC n	p-value (t-test)	p-value (npar)
Would you describe your design style as incremental?	4.18	1.76	203	4.90	1.65	187	0.000**	0.000**

Comments: — Response scale: 1 : no, not at all ; 7 : yes, absolutely.
— P-value: ** = p<0.01; * = p<0.025; npar = Mann-Witney-U test.

Table VI.5. Comparing the respondents' design styles.

In summary, the data supports that both responding groups have a high degree of design and simulation experience and thus are qualified to respond to my questions. Furthermore, EPLD and ASIC tend to see themselves as users of incremental design strategies which suggests that much of their learning happens by iterative trial and error, or in more general terms, by *experimentation*.

VI.4.2 Findings Related to Information Available *Prior* to Starting Experimentation

In this section, I will examine information that is available to designers prior to starting experimentation in a particular design project. More specifically, I will be able to test hypotheses (or parts of hypotheses) that do not depend on information from experimental cycles. The following hypotheses will be tested (the parts to be tested are underlined):

Hypothesis 1: Experimentation modes (e.g. simulation, prototype testing) vary with respect to the efficiency of conducting the four experimental steps within an experimental cycle.

Explanation: Given that simulation and prototype testing vary with respect to the steps within an experimental cycle and that some of this variation is independent of repeated experimentation, I propose that such variation will be reflected in the designers' mode evaluations prior to starting experimentation.

Hypothesis 3: Changes in the model accuracy (*ceteris paribus*) causes a movement of the experimental mode's efficiency trajectory and, as a result, a shift in the optimal mode switching point (OSP). Thus, in the case of switching between simulation and prototype testing, accuracy improvements in the former will shift the OSP away from the latter.

Explanation: We have seen in the first field study that the simulation of error-related behavior can be conducted with three different experimentation activities: the elimination of (1) logic errors; (2) timing errors; and (3) signal quality errors. Thus, designers typically use three different simulation tools ("modes") that are designed to deal with each particular class of errors: (1) logic simulation tools; (2) timing simulation tools; and (3) simulation tools related to signal quality. We further found that these three tool sets differ significantly with respect to their accuracy in modeling error-related behavior. As a result, I hypothesize that the three efficiency trajectories are

shifted relative to each other which is reflected in their relative efficiencies prior to starting simulation (see illustration in figure VI.7). Furthermore, I propose that this difference is reflected in the *a priori* selection of the three simulation "modes" relative to prototype testing.

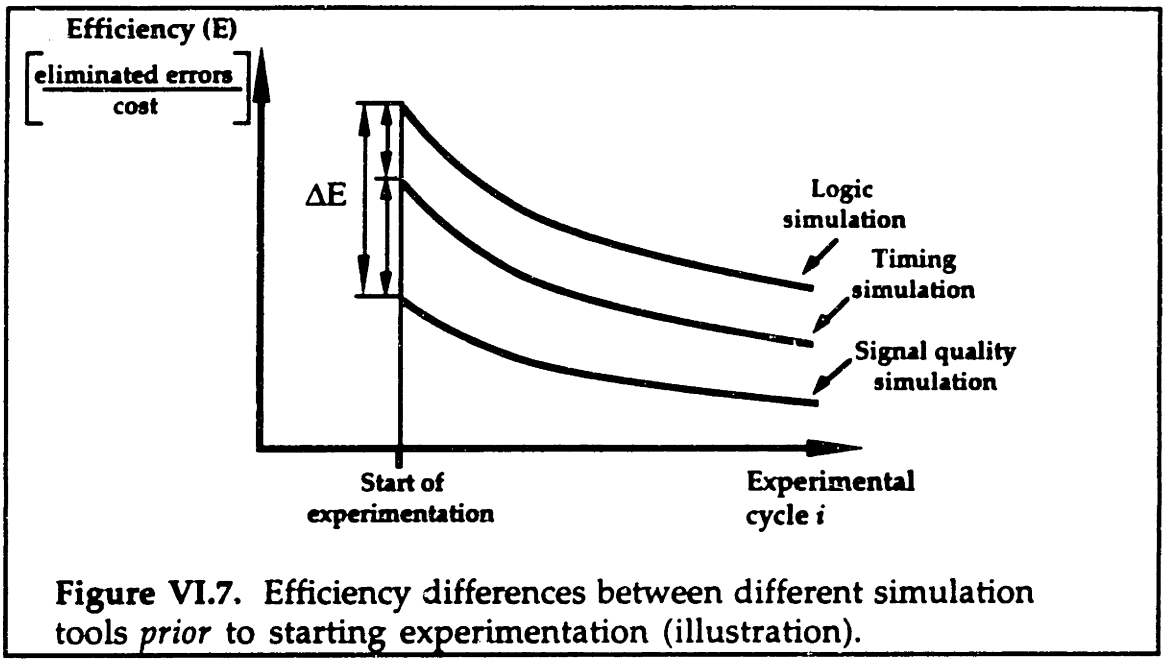


Figure VI.7. Efficiency differences between different simulation tools *prior* to starting experimentation (illustration).

Hypothesis 4: Changes in the cost to build (and/or modify) a model (*ceteris paribus*) causes a movement of the experimental mode's efficiency trajectory and, as a result, a shift in the optimal mode switching point (OSP). Thus, in the case of switching between simulation and prototype testing, cost reductions in the latter will shift the OSP away from the former.

Explanation: Since I have selected to study two groups of designers (EPLD and ASIC) who face very different costs to build (and/or modify) experimentation models, I propose that

their prototype efficiency trajectories are also shifted relative to each other. This would imply that the efficiency of simulation relative to prototype testing is significantly different for both groups prior to starting experimentation (see illustration in figure VI.8). Furthermore, I propose that this difference is reflected in the *a priori* selection of simulation over prototype testing.

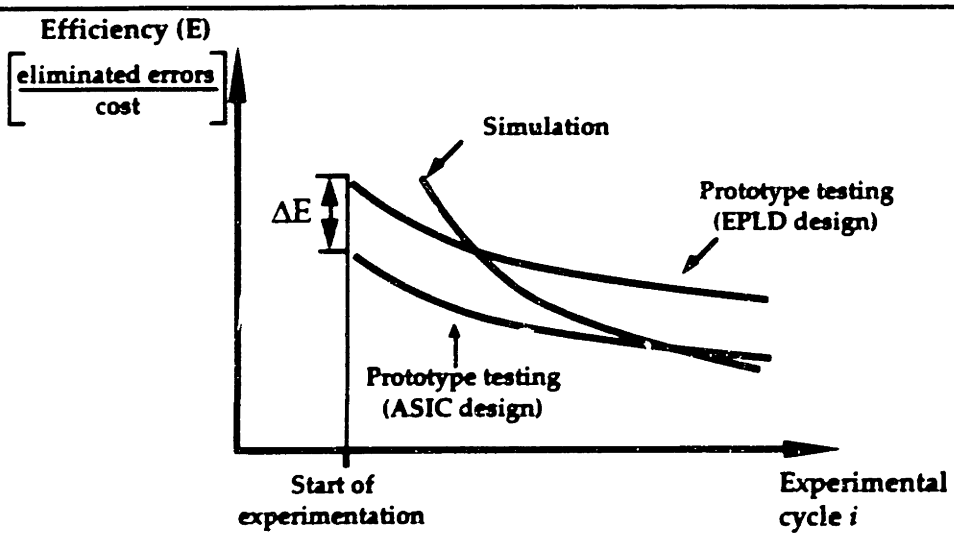


Figure VI.8. Efficiency differences between EPLD and ASIC designs *prior* to starting experimentation (illustration).

Hypothesis 5: Prior to starting experimentation, designers have very limited knowledge of the modes' efficiency trajectories. Thus, as a design progresses, they will iteratively "learn" about mode efficiencies from the experiments conducted.

VI.4.2.1 Mode Effectiveness as a Function of Experimental Steps and Error Classes

In section II of the questionnaire, I asked both responding groups (EPLD and ASIC designers) to rate the general effectiveness of modes (computer simulation and hardware prototype testing) independent of a particular design project (i.e. prior to starting experimentation) but as a function of experimental steps and error classes. Thus, I can compare their responses as (1) a function of design group (EPLD and ASIC); (2) as a function of experimental steps (detect, analyze, and fix&verify); and (3) as a function of error class (logic, timing, and signal quality).

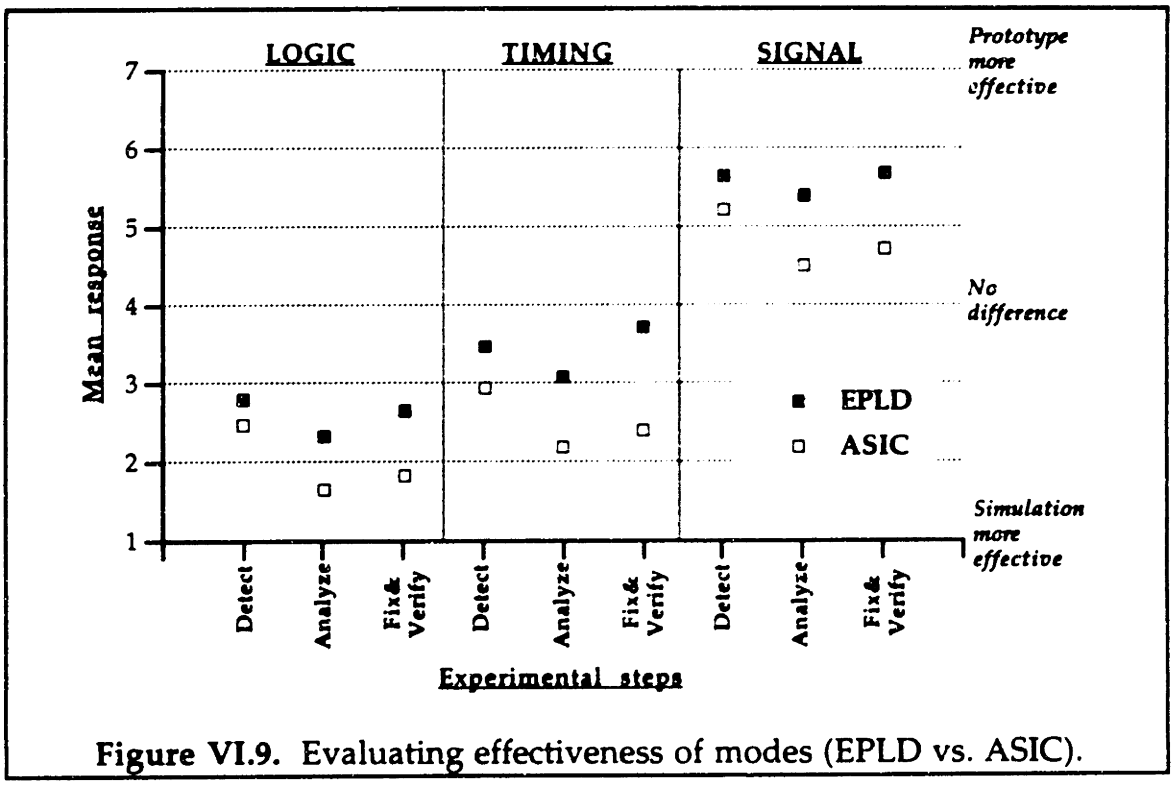


Figure VI.9. Evaluating effectiveness of modes (EPLD vs. ASIC).

I start with (1). The average responses are depicted in figure VI.9 and the results of a statistical comparison is presented in table VI.6. The

comparison between EPLD and ASIC designers shows a significant difference ($p < 0.001$) in mode effectiveness for the experimental steps "analyze" and "fix&verify", independent of error class. Furthermore, the responses indicate that EPLD designers are generally inclined to rate prototype testing as more effective than ASIC designers do. This result is consistent with figure VI.8.

Error Class	Experimental Step	EPLD Mean	EPLD St.dev.	EPLD n	ASIC Mean	ASIC St.dev.	ASIC n	p-value (t-test)	p-value (npar)
LOGIC	Detect	2.78	2.02	180	2.49	1.77	167	0.151	0.249
	Analyze	2.34	1.49	180	1.64	1.15	167	0.000**	0.000**
	Fix & Verify	2.64	1.84	180	1.83	1.36	167	0.000**	0.000**
TIMING	Detect	3.47	2.09	180	2.94	2.11	167	0.019*	0.005**
	Analyze	3.09	1.86	180	2.20	1.52	167	0.000**	0.000**
	Fix & Verify	3.74	2.10	180	2.42	1.81	167	0.000**	0.000**
SIGNAL QUALITY	Detect	5.64	1.58	180	5.24	1.92	167	0.034	0.090
	Analyze	5.39	1.65	180	4.53	1.98	167	0.000**	0.000**
	Fix & Verify	5.69	1.53	180	4.73	1.92	167	0.000**	0.000**

Comments: — Response scale: 1 : simulation more effective; 7 : prototype more effective.
— P-value: ** = $p<0.01$; * = $p<0.025$; npar = Mann-Witney-U test.
— Only respondents who answered all 9 questions were considered.

Table VI.6. Comparing mode effectiveness between EPLD and ASIC designs.

The comparison also shows that the EPLD-ASIC difference for the step "detect" is not significant in the case of logic and signal quality errors and somewhat less significant in the case of timing errors. An examination of the original question that relates to "detect" may help in finding an explanation:

"Which method is more effective in detecting all [...] errors if you could run all possible test cases (i.e. which method is more accurate)?"

Recall that a design error remains undetected for two reasons: (1) the simulation model lacks accuracy with respect to "reality"; or (2) an incomplete set of test vectors is run because exhaustive testing is infeasible. The "detect" question gives the respondent a hypothetical situation where exhaustive testing is possible and therefore it emphasizes the accuracy of simulation and prototype testing (reason (1)). Since physical behavior related to timing and signal quality is more difficult to model than logic behavior, it is reasonable to expect the difference between EPLD and ASIC to reflect such variation[14].

Result 1: When comparing EPLD and ASIC designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for most error classes and experimental steps. Overall, EPLD designers tend to rate the relative effectiveness of prototype testing over simulation higher than ASIC designers do. This result is in support of hypothesis 4.

Mode Effectiveness as a Function of Experimental Steps (EPLD Only)

I now compare mode effectiveness as a function of experimental steps ("detect", "analyze", and "fix&verify") (table VI.7 on the next page). A statistical comparison results in significant differences for the following steps: (1) "detect" versus "analyze"; and (2) "analyze" versus "fix&verify" ($p < 0.01$ for all error classes). Thus simulation is most effective relative to prototype testing in error analysis if compared to the effectiveness in error detection

[14] In other words, EPLD and ASIC designers equally find simulation models to be very effective for logic behavior and to be very ineffective for behavior related to signal quality (no significant difference). They disagree on the effectiveness of simulating timing behavior which is probably due to the difference in tools available to both groups (significant difference).

("detect") or correction ("fix & verify"). Considering the advantages that computer simulation has in problem analysis, the finding is not surprising.

In contrast, the difference between the steps "detect" and "fix&verify" is not statistically significant. This result is consistent with my general understanding of the economics of EPLD design. As designers perceive the cost of fixing errors to be low (as opposed to ASICs where it is high), they tend to be more concerned with the effectiveness of verification rather than the cost of fixing errors. But similar to detection, the effectiveness of verification depends on the accuracy of the experimentation model and therefore one would expect an insignificant difference between "detect" and "fix&verify" (in contrast, one would expect the difference to be significant for ASIC designers).

Error Class	Comparison	Paired Diff.	Std. Dev.	n	p-value (t-test)	p-value (npar)
LOGIC	(Detect) - (Analyze)	0.44	2.15	180	0.006**	0.005**
	(Detect) - (Fix&Verify)	0.14	2.07	180	0.351	0.321
	(Analyze) - (Fix&Verify)	-0.30	1.45	180	0.006**	0.005**
TIMING	(Detect) - (Analyze)	0.38	1.93	180	0.008**	0.011*
	(Detect) - (Fix & Verify)	-0.27	1.86	180	0.056	0.056
	(Analyze) - (Fix &Verify)	-0.65	1.46	180	0.000**	0.000**
SIGNAL QUALITY	(Detect) - (Analyze)	0.25	1.15	180	0.004**	0.003**
	(Detect) - (Fix&Verify)	-0.05	1.15	180	0.559	0.448
	(Analyze) - (Fix&Verify)	-0.30	1.02	180	0.000**	0.000**

Comments: — Response scale: 1 : simulation more effective; 7 : prototype more effective.
— P-value: ** = $p<0.01$; * = $p<0.025$; npar = Wilcoxon signed rank test.
— Only respondents who answered all 9 questions were considered.

Table VI.7. Comparing the mode effectiveness between experimental steps (EPLD).

Result 2: For EPLD designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for most experimental steps. The relative effectiveness of simulation over prototype testing tends to be highest for the step "analyze" when compared to "detect" and "fix&verify". The only insignificant difference (between "detect" and "fix&verify") is consistent with the economics of EPLD design. This result is in support of hypothesis 1.

Mode Effectiveness as a Function of Error Type (EPLD)

An inspection of EPLD designers' average responses (figure VI.10 on the next page) reveals the following pattern: the effectiveness of prototype testing relative to simulation increases with error class (using the following error class sequence: logic → timing → signal quality). This observation is consistent with my hypothesis that the respective simulation tools differ in accuracy and thus they have different efficiency trajectories (which was illustrated in figure V.7).

This observation is further confirmed by a rigorous statistical comparison of response distributions (table VI.8 on the next page). A statistical comparison of effectiveness results in differences that are significant ($p<0.001$) for all error classes (logic errors, timing errors, and signal quality errors) in all experimental steps considered (detect, analyze, fix&verify). Furthermore, the paired differences are all negative which implies that the rated effectiveness of prototype testing relative to simulation increases with error class.

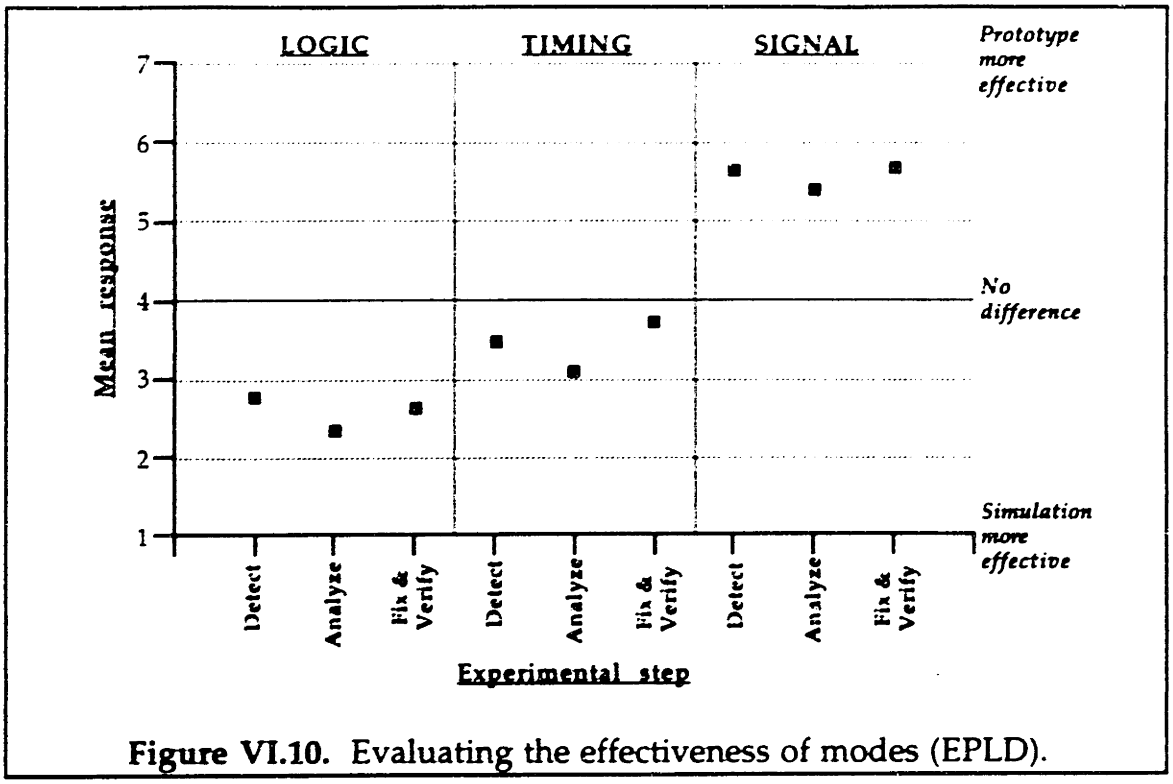


Figure VI.10. Evaluating the effectiveness of modes (EPLD).

Experimental Step	Comparison	Paired Diff.	Std. Dev.	n	p-value (t-test)	p-value (npar)
DETECT	(Logic) - (Timing)	-0.69	2.50	180	0.000**	0.000**
	(Logic) - (Signal Quality)	-2.86	2.51	180	0.000**	0.000**
	(Timing) - (Signal Quality)	-2.17	2.26	180	0.000**	0.000**
ANALYZE	(Logic) - (Timing)	-0.75	1.88	180	0.000**	0.000**
	(Logic) - (Signal Quality)	-3.06	2.13	180	0.000**	0.000**
	(Timing) - (Signal Quality)	-2.31	2.22	180	0.000**	0.000**
FIX & VERIFY	(Logic) - (Timing)	-1.10	2.30	180	0.000**	0.000**
	(Logic) - (Signal Quality)	-3.06	2.30	180	0.000**	0.000**
	(Timing) - (Signal Quality)	-1.96	2.34	180	0.000**	0.000**

Comments: — Response scale: 1 : simulation more effective; 7 : prototype more effective.
— P-value: ** = $p<0.01$; * = $p<0.025$; npar = Wilcoxon signed rank test.
— Only respondents who answered all 9 questions were considered.

Table VI.8. Comparing the mode effectiveness between error classes (EPLD).

Result 3: For EPLD designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for all error classes. Overall, the relative effectiveness of prototype testing over simulation increases with error class (logic → timing → signal quality). This result is in support of hypothesis 3.

Mode Effectiveness as a Function of Experimental Steps (ASIC Only)

Comparing the effectiveness of simulation and prototype testing as a function of experimental steps ("detect", "analyze", and "fix&verify"), we find a pattern that differs only slightly from the findings in EPLD designs (table VI.9).

Error Class	Comparison	Paired Diff.	Std. Dev.	n	p-value (t-test)	p-value (npar)
LOGIC	(Detect) - (Analyze)	0.86	1.96	167	0.000**	0.000**
	(Detect) - (Fix&Verify)	0.67	1.89	167	0.000**	0.000**
	(Analyze) - (Fix&Verify)	-0.19	0.98	167	0.013*	0.014*
TIMING	(Detect) - (Analyze)	0.74	1.92	167	0.000**	0.000**
	(Detect) - (Fix & Verify)	0.52	1.91	167	0.001**	0.000**
	(Analyze) - (Fix &Verify)	-0.22	1.14	167	0.013*	0.014*
SIGNAL QUALITY	(Detect) - (Analyze)	0.71	1.74	167	0.000**	0.000**
	(Detect) - (Fix&Verify)	0.52	1.50	167	0.000**	0.000**
	(Analyze) - (Fix&Verify)	-0.19	1.47	167	0.094	0.121

Comments: — Response scale: 1 : simulation more effective; 7 : prototype more effective.
— P-value: ** = p<0.01; * = p<0.025; npar = Wilcoxon signed rank test.
— Only respondents who answered all 9 questions were considered.

Table VI.9. Comparing the mode effectiveness between experimental steps (ASIC).

A statistical comparison results in significant differences between all steps, except for one case: the difference between the steps "analyze" and

"fix&verify" is not significant in the elimination of signal quality errors. (It is significant in the elimination of logic and timing errors.)

Also recall that the difference between "detect" and "fix&verify" was not significant in EPLD designs but it is significant in ASIC designs. This reinforces my earlier explanation that the perceived cost of fixing an error has a strong influence on assessing the effectiveness of the "fix&verify" step.

Result 4: For ASIC designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for most experimental steps. The relative effectiveness of simulation over prototype testing tends to be highest for the step "analyze" when compared to "detect" and "fix&verify". Again, this result is in support of hypothesis 1.

Mode Effectiveness as a Function of Error Type (ASIC Only)

An inspection of ASIC designers' average responses reveals the same pattern that was identified in the case of EPLD designers. Mode effectiveness varies as a function error class and the effectiveness of prototype testing relative to simulation increases with error class (logic → timing → signal quality) (figure VI.11 on the next page).

This observation is confirmed by a rigorous statistical comparison of response distributions (table VI.10 on the next page). A statistical comparison of effectiveness results in differences that are significant ($p<0.001$) for all error classes (logic errors, timing errors, and signal quality errors) in all experimental steps considered (detect, analyze, fix&verify). Furthermore, the paired differences are all negative which implies that the rated effectiveness of prototype testing relative to simulation increases with error class.

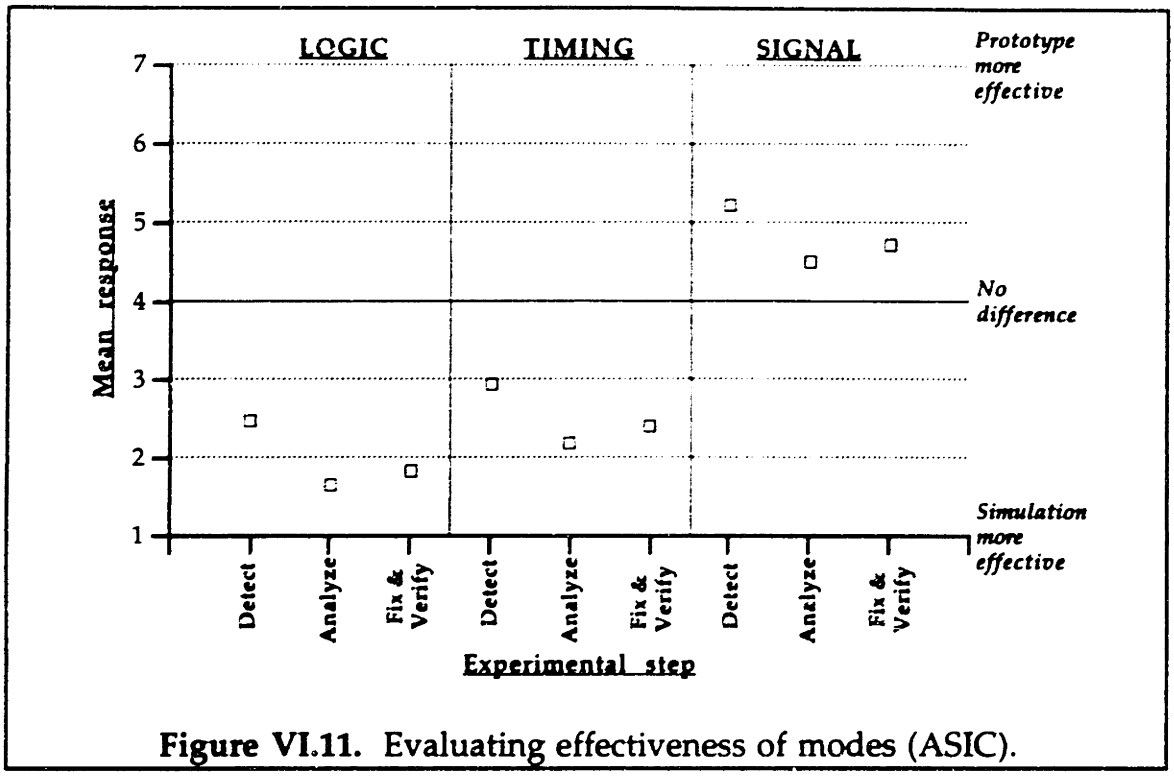


Figure VI.11. Evaluating effectiveness of modes (ASIC).

Experimental Step	Comparison	Paired Diff.	Std. Dev.	n	p-value (t-test)	p-value (npar)
DETECT	(Logic) - (Timing)	-0.45	2.27	167	0.011*	0.014*
	(Logic) - (Signal Quality)	-2.75	2.43	167	0.000**	0.000**
	(Timing) - (Signal Quality)	-2.30	2.28	167	0.000**	0.000**
ANALYZE	(Logic) - (Timing)	-0.56	1.49	167	0.000**	0.000**
	(Logic) - (Signal Quality)	-2.90	2.14	167	0.000**	0.000**
	(Timing) - (Signal Quality)	-2.34	2.11	167	0.000**	0.000**
FIX & VERIFY	(Logic) - (Timing)	-0.59	1.54	167	0.000**	0.000**
	(Logic) - (Signal Quality)	-2.90	2.21	167	0.000**	0.000**
	(Timing) - (Signal Quality)	-2.31	2.37	167	0.000**	0.000**

Comments: — Response scale: 1 : simulation more effective; 7 : prototype more effective.
— P-value: ** = p<0.01; * = p<0.025; npar = Wilcoxon signed rank test.
— Only respondents who answered all 9 questions were considered.

Table VI.10. Comparing effectiveness between error classes (ASIC).

Result 5: For ASIC designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for all error classes. Overall, the relative effectiveness of prototype testing over simulation increases with error class (logic → timing → signal quality). Again, this result is support of hypothesis 3.

VI.4.2.2 The Use of Simulation In Error Elimination

After analyzing the general effectiveness of simulation and prototype testing as a function of error class and experimental step (under *a priori* conditions), I will now examine if there is variation in the actual use of experimentation and if these choices are consistent with the earlier findings on effectiveness. In section III of the questionnaire, respondents were asked if they had used simulation for detecting errors (logic, timing, and signal quality) in their last design project. Since the decision to use simulation is made prior to starting experimentation, it must have been based on information available from previous design experience.

When comparing the responses from EPLD and ASIC designers, there appears to be a difference between the two groups in the use of simulation for eliminating timing errors and signal quality errors. The differences are found to be significant using Fisher's exact test ($p<0.001$) (see figure VI.12 and table VI.11 on the next page). In both cases, EPLD designers use simulation less frequently than ASIC designers which is consistent with the difference in net economic benefits that both groups derive from simulation. The only exception is the use of simulation in eliminating logic errors — over 90% of all respondents use logic simulation.

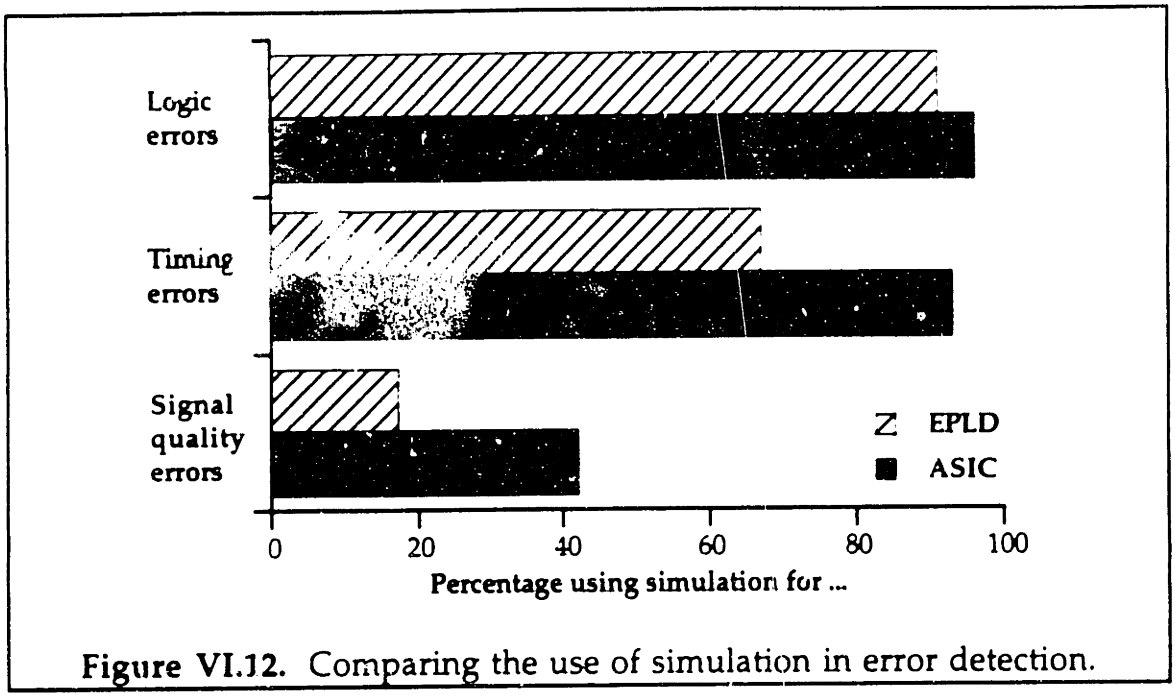


Figure VI.12. Comparing the use of simulation in error detection.

Error Class	EPLD Designers			ASIC Designers			p-value
	Use	Do not use	Total	Use	Do not use	Total	
LOGIC	180 (91.4%)	17 (8.6%)	197 (100%)	180 (96.3%)	7 (3.7%)	187 (100%)	0.058
TIMING	132 (67.0%)	65 (33.0%)	197 (100%)	173 (93%)	13 (7.0%)	186 (100%)	0.000**
SIGNAL QUALITY	34 (17.3%)	163 (82.7%)	197 (100%)	77 (42.1%)	106 (57.9%)	183 (100%)	0.000**

Comments: — P-value: ** = p<0.01; * = p<0.025; two-tailed Fisher's exact test.

Table VI.11. Number of respondents who use computer simulation in error elimination (by error class and respondent group).

Result 6: When comparing EPLD and ASIC designers, the use of simulation in experimentation differs significantly for timing and signal quality errors. No significant difference can be found for logic errors (almost all respondents in both groups use simulation). This result is in support of hypothesis 4.

Next I test for variation in the use of simulation as a function of error classes. Statistical tests for EPLD and ASIC designs are shown in table VI.12.

For EPLD designs, the use of simulation differs significantly between all error classes ($p<0.0001$). Furthermore, its relative use decreases with error class (logic → timing → signal quality).

In the case of ASIC designs, the use of simulation differs significantly between logic errors & signal quality errors and timing errors & signal quality errors. There is no significant difference in the use of simulation for logic and timing errors. Furthermore, the relative use of simulation decreases with error class (logic → timing → signal quality).

Error Class	EPLD Designers		ASIC Designers	
	Logic	Timing	Logic	Timing
Timing	0.000**		0.177	
Signal Quality	0.000**	0.000**	0.000**	0.000**

Comments: — P-value: ** = $p<0.01$; * = $p<0.025$; two-tailed Fisher's exact test.

Table VI.12. P-values in the comparison of the use of simulation.

In summary, I find that the variation in simulation use to be consistent with the [rated] mode effectiveness prior to starting experimentation (for both, EPLD and ASIC designers).

Result 7: For EPLD designs, the use of simulation in experimentation differs significantly for all error classes. Overall, the relative use of simulation decreases with error class (logic → timing → signal quality). This result is in support of hypothesis 3.

Result 8: For ASIC designs, the use of simulation in experimentation differs significantly for logic vs. signal quality errors and timing vs. signal quality errors. No significant difference can be found between logic and timing errors. (Almost all respondents used simulation in both error classes). Except for logic and timing

errors, the relative use of simulation decreases with error class (logic → timing → signal quality). Again, this result is in support of hypothesis 3.

VI.4.2.3 The State of Knowledge Prior to Starting Experimentation

A necessary condition to "learning about mode efficiencies" (hypothesis 5) is that designers' knowledge of mode efficiency trajectories is incomplete prior to starting experimentation. (If it was complete, there would be no learning and all mode switching decisions could be made prior to the start of experimentation.)

In the questionnaires, designers were asked to rate their knowledge of the relative distribution and number of design error prior to starting experimentation. (Knowing the relative distribution and number of errors is necessary to predict a mode's efficiency trajectory.) In general, designers felt that they knew something about the distribution of errors from their previous design experience[15] but they knew very little about the expected number of errors in a design (see table VI.13 on the next page). (There is no significant difference between responses from EPLD and ASIC designers.) In other words, most of their knowledge about errors (and the related mode efficiency trajectories) was expected to come from the use of computer simulation and prototype testing.

---

[15] For example, designers had completed similar designs before and were able to predict critical areas of a new design. Thus, in the design of high speed switching cards, experienced designers would be able to identify components that have a high probability of problems (prior to starting simulation).

Designers Knowledge Before Starting to Simulate [a]	EPLD Mean	EPLD St.dev.	EPLD n	ASIC Mean	ASIC St.dev.	ASIC n	p-value (t-test)	p-value (npar)
Knew type and relative distribution of errors that design would have.	3.11	1.65	193	3.35	1.70	181	0.177	0.165
Knew how many errors of each type that design would have.	2.03	1.18	193	2.09	1.42	181	0.673	0.604

Comments: — Response scale: 1 : not at all accurate ; 7 : very accurate.
— P-value: ** = p<0.01; * = p<0.025; npar = Mann-Witney-U test.
— Only respondents who answered both questions were considered.

Table VI.13. *A priori* knowledge before starting simulation.

Result 9: When comparing EPLD and ASIC designers, no significant difference in *a priori* knowledge with respect to the error distribution and the expected error count can be found.

Result 10: Both groups (EPLD and ASIC designers) had some limited knowledge about the relative distribution of errors (based on prior experience) but knew very little about the number of errors their design was expected to have. This result is in support of hypothesis 5.

VI.4.3 Findings Related to Information Available *After* Starting Experimentation

In this section, I will examine information that is available to designers after starting experimentation in a particular design project. More specifically, I will be able to test hypotheses (or parts of hypotheses) that depend on information from experimental cycles. The following hypotheses will be tested (the parts to be tested are underlined):

Hypothesis 3: Changes in the accuracy of an experimentation model (*ceteris paribus*) causes a movement of the mode's

efficiency trajectory and, as a result, a shift in the optimal mode switching point (OSP). Thus, in the case of switching between simulation and prototype testing, accuracy improvements in the former will shift the OSP away from the latter.

Explanation: In the last section, we have found some evidence supporting the hypothesis that the three simulation "modes" (logic, timing, and signal quality) have different efficiency trajectories (as a result of differences in model accuracy). To further strengthen this hypothesis, I now propose that the [optimal] mode switching point varies for these error classes which is reflected in the different switching strategies that designers use (see figure VI.13).

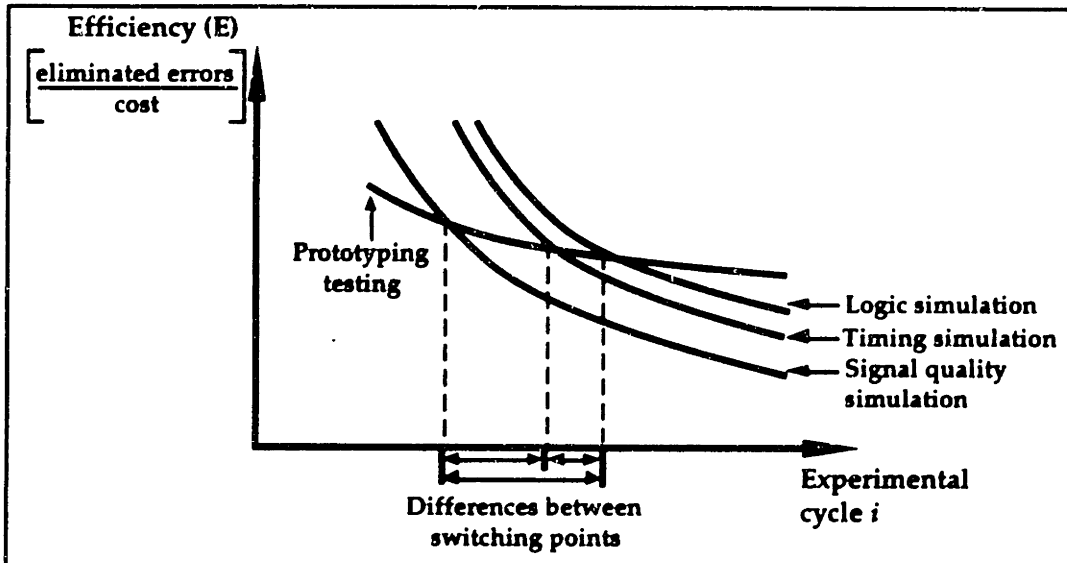


Figure VI.13. Differences in mode switching points for different simulation tools *after* starting experimentation (illustration).

Hypothesis 4: Changes in the cost to build (and/or modify) an experimentation model (*ceteris paribus*) causes a movement of the mode's efficiency trajectory and, as a result, a shift in the optimal mode switching point (OSP). Thus, in the case of switching between simulation and prototype testing, cost reductions in the latter will shift the OSP away from the former.

Explanation: In the last section, we have found some evidence supporting the hypothesis that EPLD and ASIC designers have different efficiency trajectories. To further strengthen this hypothesis, I now propose that optimal mode switching point varies for these two groups which is reflected in the different switching strategies that designers use (see figure VI.14).

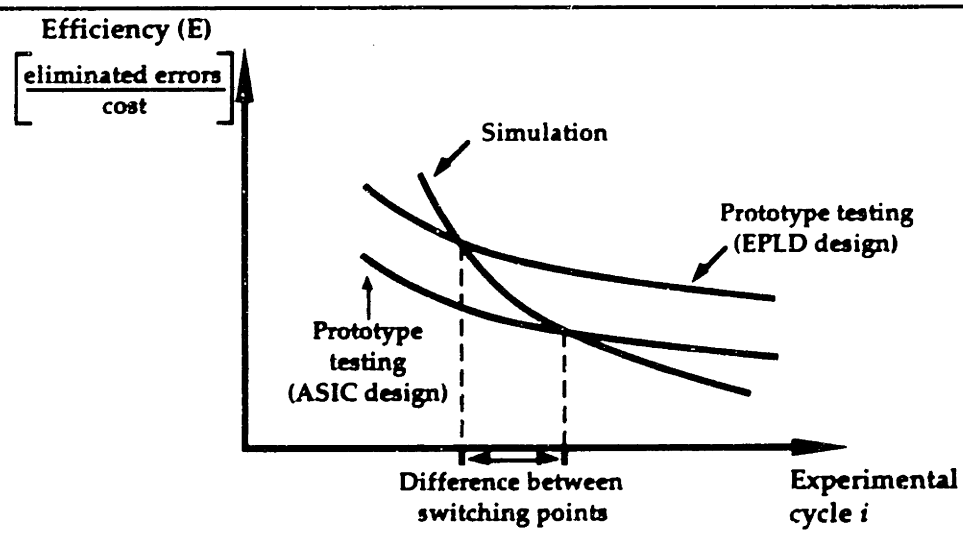


Figure VI.14. Difference in switching from simulation to prototype testing between EPLD and ASIC designs (illustration).

Hypothesis 5: Prior to starting experimentation, designers have very limited knowledge of the modes' efficiency trajectories. Thus, as a design progresses, they will iteratively "learn" about mode efficiencies from the experiments conducted.

Explanation: I have suggested earlier that designer learn about mode efficiencies by tracking the change in error elimination rates which are driven primarily by a decreasing error detection rate. Thus, instead of planning the number of experimental cycles prior to starting experimentation, I propose that designer decide whether to continue in a mode after each experimental cycle is completed and their decision is partially due to "learning about mode efficiencies" that occurs during such cycles.

VI.4.3.1 Decision Factors in Switching from Simulation to Prototype Testing

In section III of the questionnaire, I asked designers questions about the reasons why they switched from simulation to the first prototype in their last design project. The responses enabled me to divide the respondents' behavior into two categories:

— "Late" switching behavior: Designers simulate their design until they are extremely confident that no [...] errors remain. Only then they switch to experimentation with hardware prototypes.

— "Early" switching behavior: Designers switch to a hardware prototype prior to being extremely confident that no [...] errors remain in their

design. Their decision to switch "early" depends on variables that I examined as part of the questionnaire.

"Early" and "Late" Switching from Simulation to Prototype Testing

In order to separate early and late switching behavior, I first separated simulation users from non-users. Then I asked the users of simulation to rate a list of reasons for switching from computer simulation to prototype testing for the first time[16] (i.e. the first prototype).

In the first question, designers were asked to respond to the following question (the question was asked for each error class):

> "I stopped simulation because I was extremely confident that there were no remaining [...] errors in the design."

Designers who answered "yes" were categorized as "late" and those who answered "no" were categorized as "early". (The results are listed in table VI.14 on the next page.) It is interesting that about 50% of EPLD designers who use simulation (about 60% for ASIC designers) show such "early" switching behavior. An initial examination of the small difference in switching behavior between EPLD and ASIC designers is perhaps an indication that once designers are committed to simulation, there is a temptation to stay with it as long as possible, even though it may be

---

[16] In the case of ASIC designers I did not emphasize "the first time". The pretest showed that ASIC designers get confused when they are asked about the first prototype since they usually do not try more than one prototype (due to the high cost of modifying a prototype). When asked about the first time they switched, they would clearly think of the first prototype even if they had prototyped more than once.

uneconomical to do so (this is especially true for EPLD designers who switch "late" but may be better off switching "early")[17].

Error Class	EPLD Designers *"Extremely confident?"* Yes	No	Total	ASIC Designers *"Extremely confident?"* Yes	No	Total	p-value
LOGIC	91 (51.1%)	87 (48.9%)	178 (100%)	105 (59.3%)	72 (40.7%)	177 (100%)	0.135
TIMING	59 (45.4%)	71 54.6%)	130 (100%)	102 (61.4%)	64 (38.6%)	166 (100%)	0.007**
SIGNAL QUALITY	17 (53.1%)	15 (46.9%)	32 (100%)	38 (51.4%)	36 (48.6%)	74 (100%)	1.000

Comments: — P-value: ** = p<0.01; * = p<0.025; two-tailed Fisher's exact test.

Table VI.14. Number of respondents who were (or weren't) extremely confident that no errors remained in the design (by error class and respondent group).

Before moving to a statistical comparison of "early" and "late" switching behavior, we also need to include designers that decided to switch "very early" — that is designers who skipped simulation and went directly to prototype testing (see section VI.4.2.3). Thus I combined the datasets by adding the designers who skipped simulation to the group that showed "early" switching behavior. The final results are depicted in figure VI.15 and listed in table VI.15 (next page). The results clearly show that EPLD designers tend to switch earlier to prototype testing than ASIC designers do ($p<0.01$ for timing and signal quality errors; $p=0.051$ for logic errors).

[17] This point has significant managerial implications. Very often, management emphasizes first-time success in a prototype, irrespective of the underlying economics. But sometimes there are economic circumstances where it may be beneficial to move to prototype designs quickly and use them to eliminate design errors.

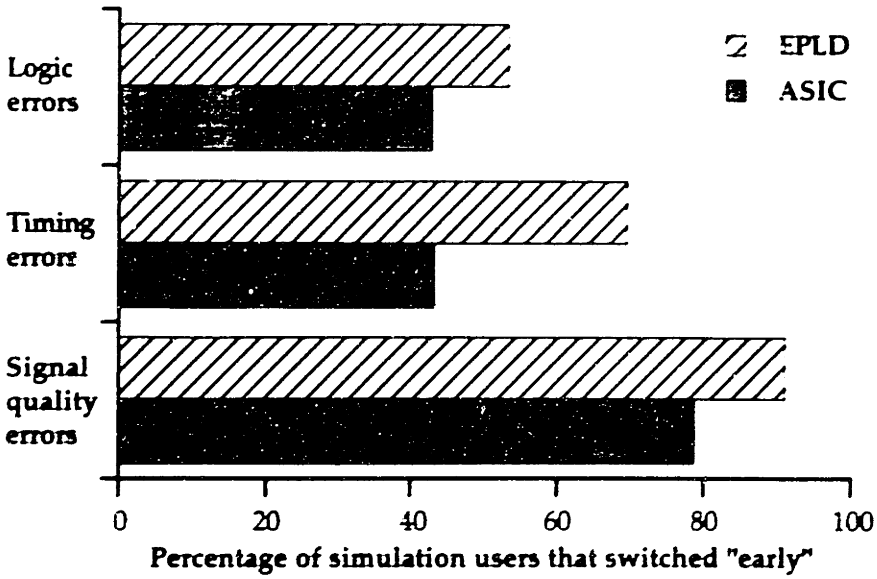


Figure VI.15. Comparison of early switching behavior (EPLD vs. ASIC).

Error Class	EPLD Designers			ASIC Designers			p-value
	Late	Early	Total	Late	Early	Total	
LOGIC	91 (46.7%)	104 (53.3%)	195 (100%)	105 (57.1%)	79 (42.9%)	184 (100%)	0.051
TIMING	59 (30.3%)	136 (69.7%)	195 (100%)	102 (57.0%)	77 (43.0%)	179 (100%)	0.000**
SIGNAL QUALITY	17 (8.7%)	178 (91.3%)	195 (100%)	38 (21.1%)	142 (78.9%)	180 (100%)	0.001**

Comments: — P-value: ** = p<0.01; * = p<0.025; two-tailed Fisher's exact test.

Table VI.15. Number of respondents who switched either "late" or "early" (by error class and respondent group).

Result 11: When comparing EPLD and ASIC designers, the difference between "early" and "late" switching behavior is significant for timing and signal quality errors ($p<0.01$). The difference is less significant for logic errors ($p=0.051$). In general, EPLD designers tend to switch to prototype testing earlier than ASIC designers do. This result is in support of hypothesis 4.

Similarly, if we compare "early" and "late" switching as a function of error class, we find that the likelihood of switching "early" increases with error class (logic → timing → signal quality) (see table VI.16). This result is consistent with my other findings so far.

Error Class	EPLD Designers		ASIC Designers	
	Logic	Timing	Logic	Timing
Timing	0.001**		1.000	
Signal Quality	0.000**	0.000**	0.000**	0.000**

Comments: — P-value: ** = p<0.01; * = p<0.025; two-tailed Fisher's exact test.

Table VI.16. P-values in the comparison of "early" and "late" switching (by error class).

Result 12: For EPLD designs, the difference between "early" and "late" switching differs significantly between all error classes. The likelihood of switching "early" increases with error class (logic → timing → signal quality). This result is in support of hypothesis 3.

Result 13: For ASIC designs, the difference between "early" and "late" switching differs significantly between logic & signal quality errors and timing & signal quality errors. It does not differ significantly between logic & timing errors. The likelihood of switching "early" increases with error class (logic → timing → signal quality). Again, this result is in support of hypothesis 3.

Decision Factors Considered in "Early" Switching Behavior (EPLD vs. ASIC)

Next I asked designers about the factors that led them to switch "early" from computer simulation to the first prototype. Respondents were asked to rate three factors[18] that played a part in their decision process and had the

[18] The three factors were determined during earlier field interviews.

opportunity to list a fourth reason if applicable. The questionnaire results are depicted in figure VI.16 and are listed in table VI.17 (next page).

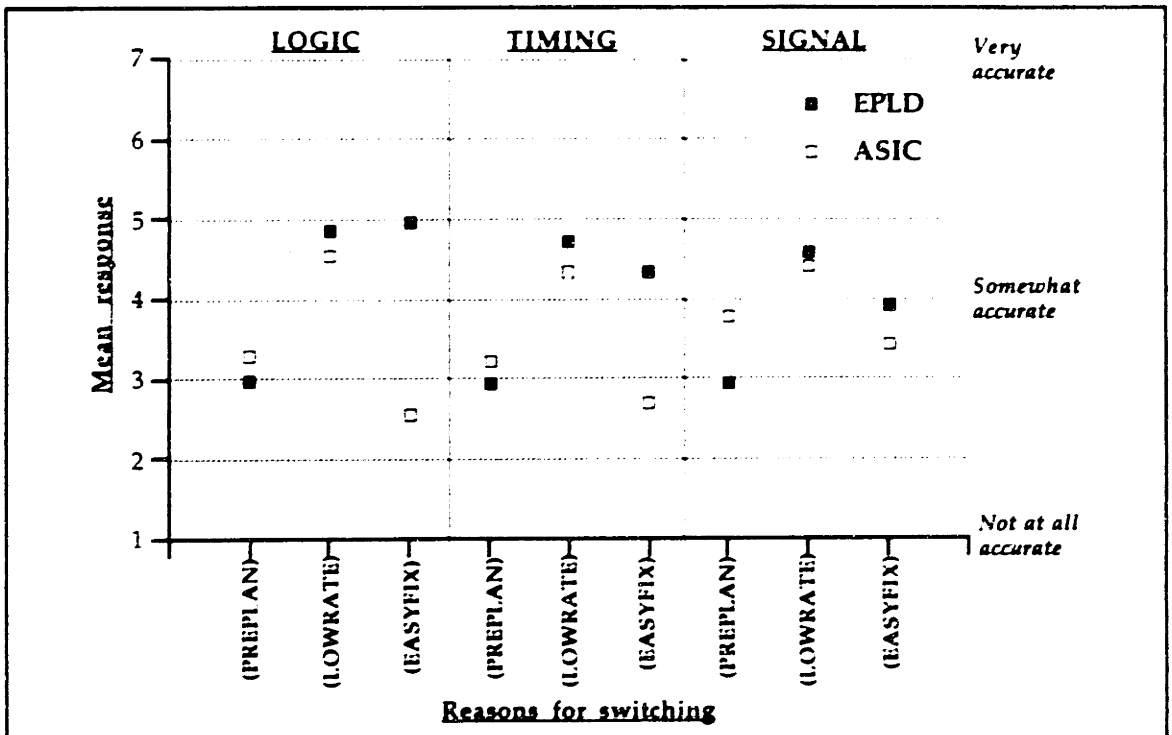


Figure VI.16. Reasons for switching (EPLD vs. ASIC).

The three factor-related questions were (coded variable first):

— (PREPLAN): "I thought that there could be remaining [...] errors but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation."

— (LOWRATE): "I thought that there could be remaining [...] errors but the rate of [...] error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining [...] errors."

— (EASYFIX): "I thought that there could be remaining [...] errors but I felt it would be easy to fix the remaining [...] errors in the hardware prototype."

A few respondents listed a fourth reason which, in most cases, indicated that they ran out of scheduled simulation time and had to move on to the next project phase (see appendix E for more information).

Error Class	Reasons for Switching	EPLD Mean	EPLD St.dev.	EPLD n	ASIC Mean	ASIC St.dev.	ASIC n	p-value (t-test)	p-value (npar)
LOGIC	(PREPLAN)	2.96	1.78	83	3.27	2.17	70	0.346	0.548
	(LOWRATE)	4.84	1.81	83	4.53	1.85	70	0.291	0.278
	(EASYFIX)	4.96	1.76	83	2.56	1.95	70	0.000**	0.000**
TIMING	(PREPLAN)	2.95	1.80	64	3.21	2.08	62	0.462	0.580
	(LOWRATE)	4.70	1.87	64	4.34	1.95	62	0.287	0.291
	(EASYFIX)	4.34	1.98	64	2.68	2.00	62	0.000**	0.000**
SIGNAL QUALITY	(PREPLAN)	2.93	1.77	14	3.79	2.01	33	0.156	0.161
	(LOWRATE)	4.57	2 10	14	4.39	1.94	33	0.789	0.609
	(EASYFIX)	3.93	2.27	14	3.42	2.06	33	0.482	0.448

Legend:

(PREPLAN): I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.

(LOWRATE): The rate of [...] error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining [...] errors.

(EASYFIX): I felt that it would be easy to fix the remaining [...] errors in the fabricated prototype.

Comments:
— Response scale: 1 : not at all accurate; 7 : very accurate.
— P-value: ** = p<0.01; * = p<0.025; npar = Mann-Witney-U test.
— Only respondents who answered all questions within an error class were considered.

Table VI.17. Comparing reasons for switching (EPLD vs. ASIC).

A comparison between EPLD and ASIC designers finds the factor (EASYFIX) to differ significantly for logic and timing errors but not for signal quality errors. This result is consistent with the cost of prototype changes which is significantly lower for EPLD-based designs. There are no significant EPLD/ASIC differences for the other decision factors.

Result 14: When comparing reasons for "early" switching between EPLD and ASIC designs, there is no significant difference for (PREPLAN) and (LOWRATE) (in all error classes) There is a significant difference for (EASYFIX) which is consistent with the relatively higher cost of prototype changes in ASIC designs. This result is in support of hypothesis 4.

Comparing Decision Factors in "Early" Switching Behavior (EPLD Only)

Finally I would like to examine the differences between the three factors that drove the decision when to switch "early" from simulation to prototype testing for EPLD designers (the results are shown in figure VI.17 and table VI.18).

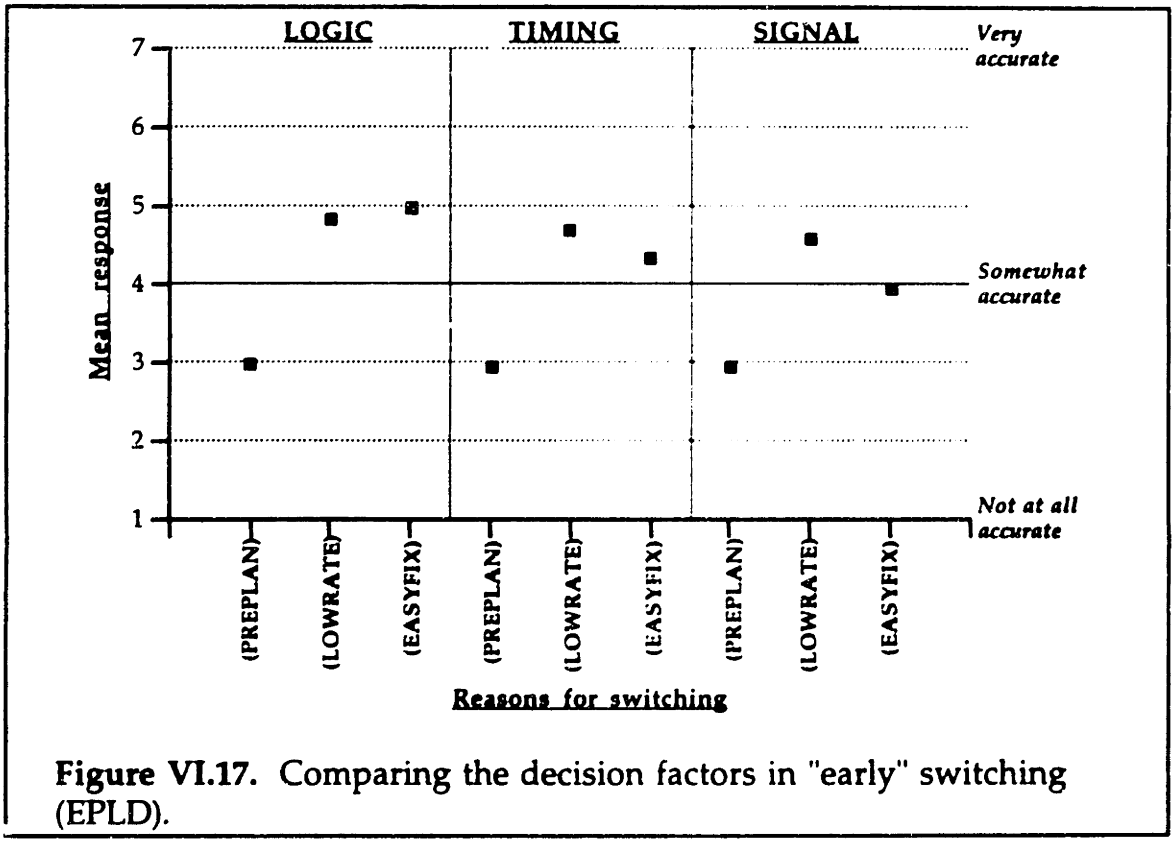


Figure VI.17. Comparing the decision factors in "early" switching (EPLD).

Error Class	Comparison	Paired Diff.	Std. Dev.	n	p-value (t-test)	p-value (npar)
LOGIC	(PREPLAN) - (LOWRATE)	-1.88	2.57	83	0.000**	0.000**
	(PREPLAN) - (EASYFIX)	-2.00	2.68	83	0.000**	0.000**
	(LOWRATE) - (EASYFIX)	-0.12	2.30	83	0.635	0.776
TIMING	(PREPLAN) - (LOWRATE)	-1.75	2.64	64	0.000**	0.000**
	(PREPLAN) - (EASYFIX)	-1.39	2.82	64	0.000**	0.001**
	(LOWRATE) - (EASYFIX)	0.36	2.26	64	0.209	0.175
SIGNAL QUALITY	(PREPLAN) - (LOWRATE)	-1.64	1.65	14	0.002**	0.005**
	(PREPLAN) - (EASYFIX)	-1.00	1.57	14	0.033	0.042
	(LOWRATE) - (EASYFIX)	0.64	1.87	14	0.220	0.202

Comments: — Response scale: 1 : not at all accurate; 7 : very accurate.
— P-value: ** = p<0.01; * = p<0.025; npar = Wilcoxon signed rank test.
— Only respondents who answered all questions within an error class were considered.

Table VI.18. Comparing influence of factors on decision-making (EPLD).

An analysis of the designers' responses results in the following findings:

— (PREPLAN) is rated significantly below (LOWRATE) as a decision factor in favor of "early" switching (for all error classes). In other words, designers cannot reasonably plan their experimentation strategy prior to starting experimentation. Instead they learn from experimental cycles; in this case by observing the incremental change in the error detection rate after completing a cycle.

— (EASYFIX) is rated fairly high but decreases with error class (logic → timing → signal quality). This implies that designers do not worry about prototype changes in the case of logic errors but the degree of concern increases with error class. This result is consistent with the findings in my first field study (chapter V). Logic errors can usually be fixed by reprogramming EPLDs (inexpensive) while signal quality errors

often require difficult changes at the hardware board level (more expensive). In summary, it is reasonable to conclude that (EASYFIX) counts in favor of switching "early".

Result 15: When comparing reasons in favor of "early" switching in EPLD designs, the data suggests that the completion of preplanned tests carries little significance while a decreasing error detection rate and the [low] cost of prototype changes are equally very significant. This result is in support of hypotheses 4 and 5.

Comparing Decision Factors in "Early" Switching Behavior (ASIC Only)

A similar analysis of ASIC designers' responses results in the following findings (see figure VI.18 and table VI.19 on the following page):

— (PREPLAN) is rated significantly below (LOWRATE) as a decision factor in favor of "early" switching (for logic and timing errors). Again, the explanation is similar to the findings in EPLD designs.

— (EASYFIX) is rated fairly low which is consistent with the high cost of prototype changes (the cost is perceived as high irrespective of error class). Thus it is reasonable to conclude that the high cost decreases the probability of switching "early".

Result 16: When comparing reasons in favor of early switching in ASIC designs, the data suggests that the completion of preplanned tests and the [high] cost of prototype changes carry little significance while a decreasing error detection rate are very significant. Again, this result is in support of hypotheses 4 and 5.

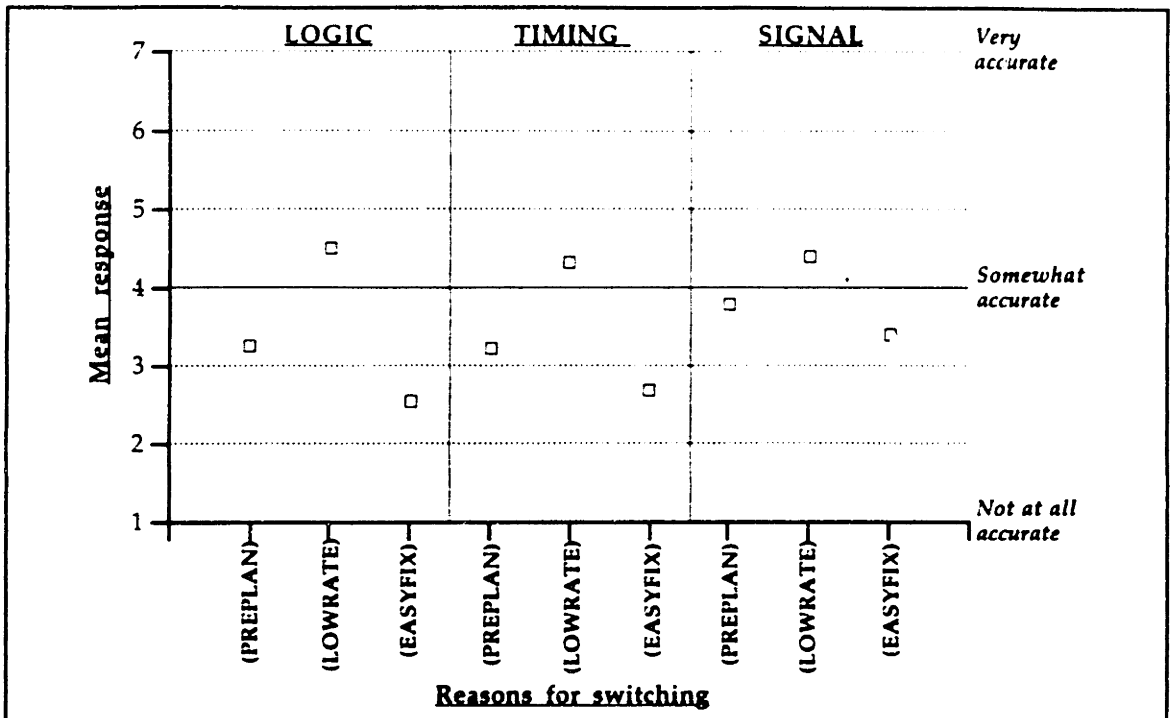


Figure VI.18. Comparing the decision factors in "early" switching (ASIC).

Error Class	Comparison	Paired Diff.	Std. Dev.	n	p-value (t-test)	p-value (npar)
LOGIC	(PREPLAN) - (LOWRATE)	-1.26	2.81	70	0.000**	0.001**
	(PREPLAN) - (EASYFIX)	0.71	3.06	70	0.055	0.059
	(LOWRATE) - (EASYFIX)	1.97	2.63	70	0.000**	0.000**
TIMING	(PREPLAN) - (LOWRATE)	-1.13	3.15	62	0.006**	0.008**
	(PREPLAN) - (EASYFIX)	0.53	2.74	62	0.132	0.108
	(LOWRATE) - (EASYFIX)	1.66	2.54	62	0.000**	0.000**
SIGNAL QUALITY	(PREPLAN) - (LOWRATE)	-0.61	3.04	33	0.261	0.166
	(PREPLAN) - (EASYFIX)	0.36	2.78	33	0.458	0.364
	(LOWRATE) - (EASYFIX)	0.97	2.95	33	0.068	0.097

Comments: — Response scale: 1 : not at all accurate; 7 : very accurate.
— P-value: ** = p<0.01; * = p<0.025; npar = Wilcoxon signed rank test.
— Only respondents who answered all questions within an error class were considered.

Table VI.19. Comparing influence of factors in decision making (ASIC).

VI.4.4 Comparing Mode Switching in Circuit Design: EPLD Versus ASIC Design Performance

While project performance was not central to my research hypotheses, I included questions (questionnaire section IV) that would provide me with enough information to include a general performance benchmark. Comparing EPLD and ASIC design performance was interesting for the following reasons:

— Proposition 2 states that as exogenous changes in model accuracy and/or model building cost lead to a shift in the optimal switching point (OSP), designers will find that adjusting their mode switching strategies in the direction of the new OSP will result it significant reductions of total design cost.

— I have seen that ASIC designers are under much pressure to "get it right in the first prototype" because of the high cost and time necessary to modify prototypes. Thus they are essentially barred from utilizing the advantages of experimentation with hardware prototypes. In contrast, EPLD designers are much less constrained with respect to the use of prototype testing as an alternative experimentation mode. Thus they can utilize the advantages of prototype testing to a much higher degree than ASIC designers can. Under proposition 2, this difference should result in a significant difference in design cost in favor of EPLD designers.

— Field interviews have suggested that EPLD design projects are completed more cost-effectively than ASIC design projects are. In addition, most EPLD vendors have also made "time-to-market" as their main core

benefit proposition, claiming that EPLD-based designs allow firms to develop products faster than using traditional ASIC technology. A close examination of the evidence revealed that most of their data was based on hearsay and that no EPLD vendor has done any formal comparisons. Thus I decided to collect data that would allow me to compare EPLD and ASIC project performance.

VI.4.4.1 Performance Measure and Project Sample

In benchmarking projects, one is usually faced with the problem of (1) finding appropriate measures of performance; and (2) comparing projects that are equivalent in complexity.

With respect to (1), I decided against using time-to-completion as a performance measure, since I believed effort (in man-months) to be more reliable. For example, effort does not include the time a design stays with an outside hardware vendor (time-to-completion does) and it is normalized with respect to design team size (time-to-completion is not). Since I am interested in measuring effort related to direct problem-solving in design, time-to-completion would have [wrongfully] included activities done by outside vendors.

With respect to (2), ASIC designs are significantly higher in complexity (as approximated by the number of gates[19]) than EPLD designs are and thus

---

[19] While size (in number of gates) is an imperfect measure of complexity, it serves as a first-level approximation and is often used by circuit designers to describe project complexity. For more elaborate discussions of complexity and its measurement, I strongly recommend the software engineering literature (e.g. Shooman 1983). For a more general discussion, work from authors associated with the Santa Fe Institute (New Mexico) provides interesting discussion on the fundamental difficulty in measuring complexity (e.g. Gell-Mann 1994, Waldrop 1993).

one would expect ASIC designs to consume more design resources. One possibility is to normalize complexity (e.g. complexity per unit effort) but such an approach is problematic because one would have to make [unproved] assumptions about the relationship between complexity and effort[20]. Instead, I decided that both samples were large enough to focus on EPLD and ASIC design projects that were roughly equivalent in size.

A close inspection of the design size distribution revealed that the region between 5 kgates and 25 kgates accounts for 30.46% of reported EPLD design projects and 25.15% of reported ASIC design projects[21]. Considering that both groups reported projects ranging from less than 0.1 kgates up to 1,000 kgates, the 5k-25kgate segment also represents an area of complexity where most of the design activities occur. Furthermore, by looking at the 5k-25k segment, one eliminates potential outliers on both ends of the complexity spectrum. As a consequence, I decided to compare projects that are located in the 5k-25kgate segment. Their histograms are shown in figures VI.19 and VI.20[22] (next page).

---

[20] For example, software designers use measures such as lines of code per man-hour in normalizing design projects (the equivalent for circuit designs would be gates per man-hour). I do not feel comfortable with such measures because they assume a linear relationship between complexity and effort.

[21] EPLD: 174 respondents reported a project size and 53 projects were between 5 kgates and 25 kgates. ASIC: 171 respondents reported a project size and 43 projects were between 5 kgates and 25 kgates.

[22] I should note that three design projects were removed from the sample used in all the comparisons because they appeared to be outliers with respect to total design effort. The removed datapoints were: (1) ASIC projects: 288 man-months (size: 25 kgates), 144 man-months (size: 25 kgates); (2) EPLD projects: 48 man-months (size: 7 kgates). Including these three projects in the test sample would have increased the difference between EPLD and ASIC projects.

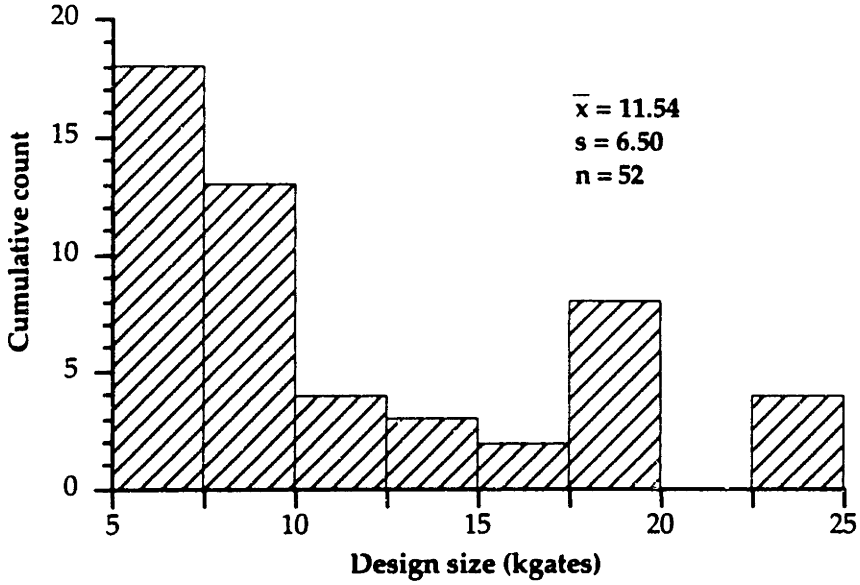


Figure VI.19. Designs between 5 k and 25 kgates (EPLD).

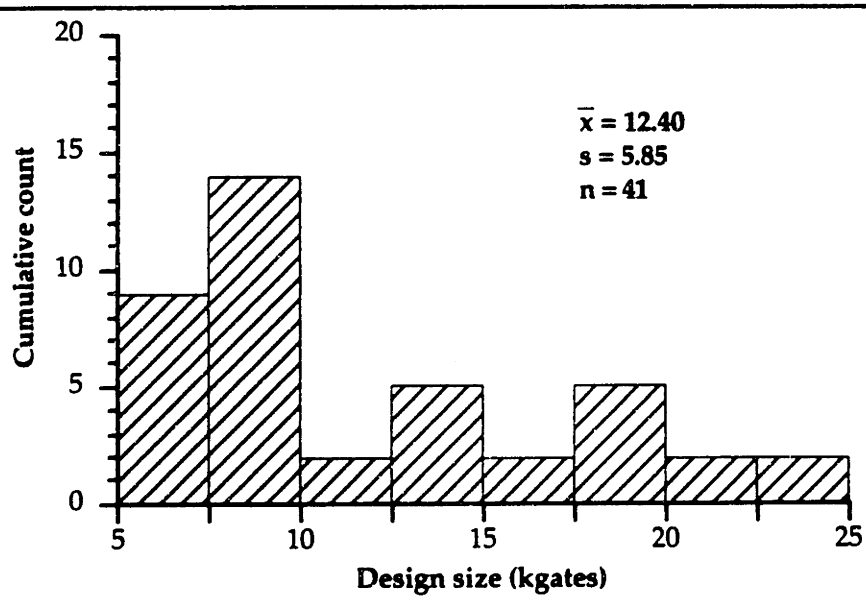


Figure VI.20. Designs between 5 k and 25 kgates (ASIC).

Within the 5k-25kgate segment, EPLD design projects have an average size of 11.54 kgates and ASIC design projects average 12.40 kgates. A

comparison of their mean project size shows that there is no significant difference between ASIC and EPLD projects (see table VI.20).

Variable	EPLD			ASIC			p -value
	Mean	St.dev.	n	Mean	St.dev.	n	(t-test)
Design size (k Gates)	11.54	6.50	52	12.40	5.85	41	0.501

Table VI.20. Comparing EPLD and ASIC design size.

Since design size is an imperfect measure of complexity, I also looked at other variables that related to design complexity and were asked in the questionnaire. The *first* variable ("percentage of design copied") accounted for the possibility that some designs have a high gate count but are of little complexity because most of the design is copied or replicated. (For example, a memory cell can be repeated many times, quickly increasing the gate count with little incremental effort.) The *second* variable ("type") captured to what degree the design was a mix between analog and digital components. If there was a significant difference in "type", one would have to analyze its impact on design complexity. The *third* variable ("has done similar designs before") measured the degree of similar design projects that designers had completed previously. The *fourth* variable ("run speed was pushed to limit") and the *fifth* variable ("high degree of logic complexity") captured rated design speed and logic complexity, respectively. In summary, respondents provided six measures (design size and the five measures descried above) that all relate to design complexity. A comparison of means between EPLD and ASIC design projects shows no significant difference ($p < 0.01$) for any of the five measures (see table VI.21). (Table VI.20 showed no significant difference in design size.)

Variable	EPLD Mean	EPLD St.dev.	EPLD n	ASIC Mean	ASIC St.dev.	ASIC n	p-value (t-test)	p-value (npar)
Percentage of design copied (%)	26.08	21.15	51	28.66	26.38	38	0.622	—
Type (1=analog; 7=digital)	5.92	1.03	52	6.10	1.46	41	0.519	0.086
Has done similar designs before	4.31	1.79	52	4.61	1.91	41	0.438	0.407
Run speed was pushed to limit	4.33	1.87	52	4.29	2.19	41	0.937	0.994
High degree of logic complexity	5.02	1.32	52	4.27	1.73	41	0.024*	0.035

Comments: — Response scale: 1 : not at all accurate ; 7 : very accurate.
— P-value: ** = p<0.01; * = p<0.025; npar = Mann-Witney-U test.

Table VI.21. Comparing EPLD and ASIC design projects (gate count between 5 kgates and 25 kgates).

Thus it is reasonable to assume that projects from both groups are of similar complexity and that an objective comparison of project performance can be conducted.

Result 17: Sampled EPLD and ASIC design projects with gate counts between 5 kgates and 25 kgates do not differ significantly with respect to gate count, novelty, signal (analog vs. digital), rated speed and rated logic complexity.

Finally I take a look at how recent the compared design projects were completed (table VI.22 on the following page). EPLD designs were completed an average 6.55 months ago while ASIC designs were completed an average 12.32 months ago. The difference is not significant at $p<0.025$. (It is significant at $p<0.05$.) As we will see later in this section, a small difference in recency is no surprise as ASIC design projects take significantly more effort to develop than EPLD design projects (in man-months). However, design projects from

both groups can be considered as relatively recent and a minor difference does not raise any concern with respect to comparing them.

Variable	EPLD			ASIC			p-value
	Mean	St.dev.	n	Mean	St.dev.	n	(t-test)
Time since completion (months)	6.55	7.90	51	12.32	14.80	41	0.028

Table VI.22. Time since design project completion.

VI.4.4.2 Comparing Project Performance and Prototype Iterations

An analysis of design effort shows that EPLD designers needed an average effort of 8.45 man-months to complete their projects. In contrast, ASIC designers needed 19.24 man-months which is by a factor of 2.28 higher than the effort that EPLD designers needed. The distribution of design effort for both groups is shown in figures VI.21 and VI.22.

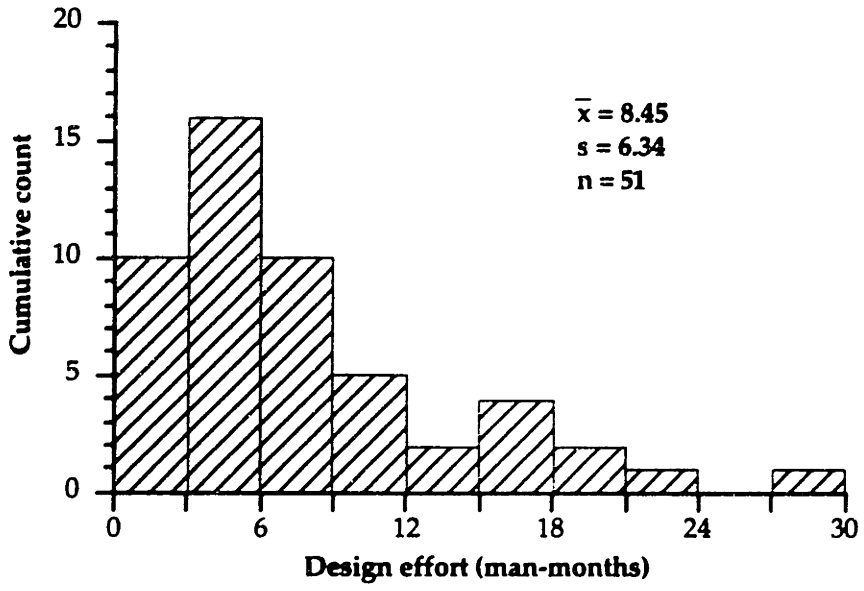


Figure VI.21. Design effort between 5 and 25 kgates (EPLD).

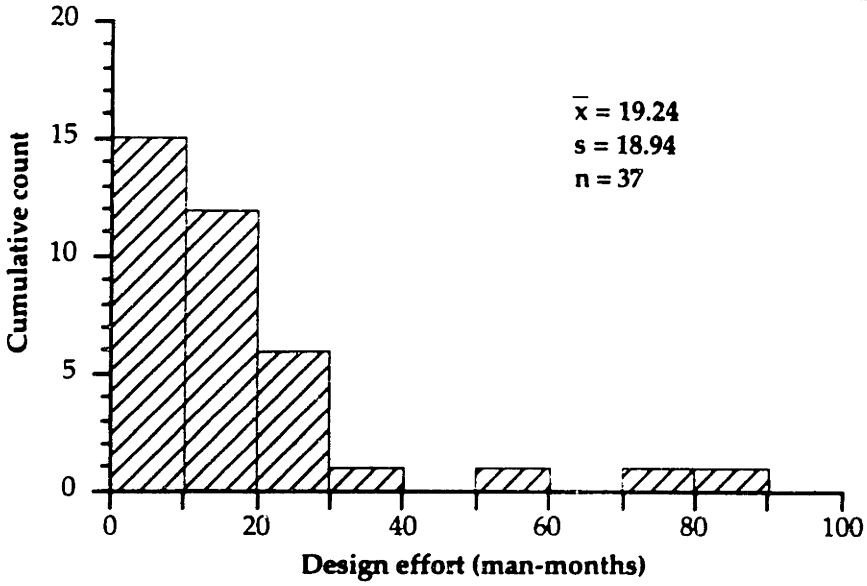


Figure VI.22. Design effort between 5k and 30 kgates (ASIC) (please note that the horizontal scale runs to 100 man-months, as opposed to 30 man-months in figure VI.21).

As expected from an inspection of their histograms, a comparison of means between EPLD and ASIC design projects shows the difference in design effort to be statistically significant ($p < 0.01$) (see table VI.23 on the following page). Besides providing a figure for the total design effort needed in completing their project, designers were also asked to estimate the percentage of effort that was due to the following design phases: design specification, design development, design verification, design prototyping, and design evaluation (see appendix B for a description of these phases). The results are listed in table VI.23 and mean values are plotted in figure VI.23 (next page).

Variable	EPLD Mean	EPLD St.dev.	EPLD n	ASIC Mean	ASIC St.dev.	ASIC n	p -value (t-test)
Design effort (man-months)	**8.45**	**6.34**	**51**	**19.24**	**18.94**	**37**	**0.002****
- % due to design specification	15.74	10.62	50	16.09	7.89	28	0.869
- % du e to design development	33.76	13.69	50	30.54	13.26	28	0.313
- % due to design verification	**17.92**	**8.06**	**50**	**28.63**	**13.48**	**28**	**0.000****
- % due to design prototyping	14.35	9.33	50	12.08	12.85	28	0.417
- % due to design evaluation	18.43	10.94	50	14.02	10.39	28	0.083

Comments: — Response scale: 1 : not at all accurate ; 7 : very accurate.
— P-value: ** = p<0.01; * = p<0.025.
— Only respondents who answered all 5 (- % due to ...) questions. were included in the design phase analysis.

Table VI.23. Comparing design effort and design phases between EPLD and ASIC projects (5k - 25 kgates).

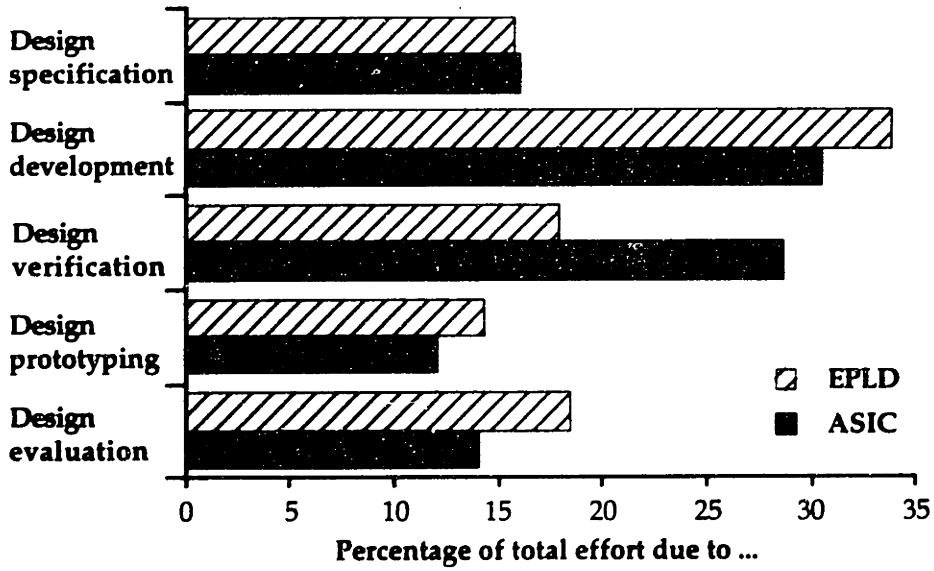


Figure VI.23. Comparing design phases between EPLD and ASIC design projects.

A close examination of the design phases shows that there is no significant difference for design specification, design development, design

prototyping[23], and design evaluation. A comparison of the percentages due to design verification, however, results in a significant difference ($p < 0.001$). Design verification involves mostly simulation and EPLD designers appear to invest less relative effort in it. This result is in line with my general argument that EPLD designer simulate less than ASIC designers.

Result 18: Sampled EPLD and ASIC design projects with gate counts between 5 kgates and 25 kgates differ significantly with respect to total required effort (man-months) and design phase distribution. In the sample, EPLD design projects are completed with significantly less effort than ASIC design projects (EPLD: $\bar{x}=8.45$ and $s=6.34$ man-months; ASIC: $\bar{x}=19.24$ and $s=18.94$ man-months). Furthermore, ASIC designers spent significantly more relative effort on design verification. This result is in support of proposition 2.

Prototype Iterations

An analysis of the use of prototyping in experimentation results in a rather dramatic contrast: on the average, EPLD designers used 13.90 prototype iterations[24] before the design was complete while ASIC designers used 1.49 prototype iterations (the difference is significant at $p < 0.001$) (see table VI.24 on the next page). This result strongly supports my proposition that EPLD designers incorporate prototype testing in their experimentation strategy to a

---

[23] In ASIC design, prototyping can take several weeks and may delay time-to-market significantly. But since prototyping is often done by outside foundries, most respondents were not expected to include the actual ASIC fabrication/customization effort in their reported figures. In contrast, EPLD designers often customize prototypes themselves.

[24] A prototype iteration is defined as: "A prototype iteration occurs whenever the designer makes a change to any part of the physical design prototype and subsequently verifies it."

much higher degree than ASIC designers do (and, as a result, switch to prototype testing much earlier).

Variable	EPLD			ASIC			p -value
	Mean	St.dev.	n	Mean	St.dev.	n	(t-test)
Number of prototype iterations	**13.90**	**14.77**	**51**	**1.49**	**1.48**	**33**	**0.000****
- % due to design errors	49.30	32.00	49	31.38	35.73	21	0.055
- % due to specification changes	30.97	27.69	49	26.10	34.14	21	0.567
- % due to other reasons	19.71	27.16	49	42.48	45.37	21	0.042

Comment: — P-value: ** = p<0.01; * = p<0.025.

Table VI.24. Comparing prototype iterations of EPLD and ASIC design projects (between 5 k and 25 kgates).

Result 19: Sampled EPLD and ASIC design projects with gate counts between 5 kgates and 25 kgates differ significantly with respect to the number of prototype iterations used until design completion (EPLD: $\bar{x}$=13.90 and s=14.77; ASIC: $\bar{x}$=1.49 and s=1.48). This result is in support of hypothesis 4.

VI.5 Conclusion

In this chapter, I used the findings from a large-scale survey to support the hypotheses structure developed in chapter III. (A summary of the results are shown in tables VI.25a and VI.25b which are on the next two pages.) Given the high degree of statistical significance for most analyses and the general consistency of my findings with the hypotheses structure and earlier field research, I conclude that this second field study has been very successful.

No.	Result Content — (as found in main text)
1	When comparing EPLD and ASIC designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for most error classes and experimental steps. Overall, EPLD designers tend to rate the relative effectiveness of prototype testing over simulation higher than ASIC designers do. This result is in support of hypothesis 4.
2	For EPLD designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for most experimental steps. The relative effectiveness of simulation over prototype testing tends to be highest for the step "analyze" when compared to "detect" and "fix&verify". The only insignificant difference (between "detect" and "fix&verify") is consistent with the economics of EPLD design. This result is in support of hypothesis 1.
3	For EPLD designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for all error classes. Overall, the relative effectiveness of prototype testing over simulation increases with error class (logic → timing → signal quality). This result is in support of hypothesis 3.
4	For ASIC designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for most experimental steps. The relative effectiveness of simulation over prototype testing tends to be highest for the step "analyze" when compared to "detect" and "fix&verify". Again, this result is in support of hypothesis 1.
5	For ASIC designs, the effectiveness of simulation and prototype testing in experimentation differs significantly for all error classes. Overall, the relative effectiveness of prototype testing over simulation increases with error class (logic → timing → signal quality). Again, this result is support of hypothesis 3.
6	When comparing EPLD and ASIC designers, the use of simulation in experimentation differs significantly for timing and signal quality errors. No significant difference can be found for logic errors (almost all respondents in both groups use simulation). This result is in support of hypothesis 4.
7	For EPLD designs, the use of simulation in experimentation differs significantly for all error classes. Overall, the relative use of simulation decreases with error class (logic → timing → signal quality). This result is in support of hypothesis 3.
8	For ASIC designs, the use of simulation in experimentation differs significantly for logic vs. signal quality errors and timing vs. signal quality errors. No significant difference can be found between logic and timing errors. (Almost all respondents used simulation in both error classes). Except for logic and timing errors, the relative use of simulation decreases with error class (logic → timing → signal quality). Again, this result is in support of hypothesis 3.
9	When comparing EPLD and ASIC designers, no significant difference in *a priori* knowledge with respect to the error distribution and the expected error count can be found.
10	Both groups (EPLD and ASIC designers) had some limited knowledge about the relative distribution of errors (based on prior experience) but knew very little about the number of errors their design was expected to have. This result is in support of hypothesis 5.

Table VI.25a. Summary of results 1-10 (continued on table VI.25b).

No.	Result Content — (continued)
11	When comparing EPLD and ASIC designers, the difference between "early" and "late" switching behavior is significant for timing and signal quality errors ($p<0.01$). The difference is less significant for logic errors ($p=0.051$). In general, EPLD designers tend to switch to prototype testing earlier than ASIC designers do. This result is in support of hypothesis 4.
12	For EPLD designs, the difference between "early" and "late" switching differs significantly between all error classes. The likelihood of switching "early" increases with error class (logic → timing → signal quality). This result is in support of hypothesis 3.
13	For ASIC designs, the difference between "early" and "late" switching differs significantly between logic & signal quality errors and timing & signal quality errors. It does not differ significantly between logic & timing errors. The likelihood of switching "early" increases with error class (logic → timing → signal quality). Again, this result is in support of hypothesis 3.
14	When comparing reasons for "early" switching between EPLD and ASIC designs, there is no significant difference for (PREPLAN) and (LOWRATE) (in all error classes) There is a significant difference for (EASYFIX) which is consistent with the relatively higher cost of prototype changes in ASIC designs. This result is in support of hypothesis 4.
15	When comparing reasons in favor of "early" switching in EPLD designs, the data suggests that the completion of preplanned tests carries little significance while a decreasing error detection rate and the [low] cost of prototype changes are equally very significant. This result is in support of hypotheses 4 and 5.
16	When comparing reasons in favor of early switching in ASIC designs, the data suggests that the completion of preplanned tests and the [high] cost of prototype changes carry little significance while a decreasing error detection rate are very significant. Again, this result is in support of hypotheses 4 and 5.
17	Sampled EPLD and ASIC design projects with gate counts between 5 kgates and 25 kgates do not differ significantly with respect to gate count, novelty, signal (analog vs. digital), rated speed and rated logic complexity.
18	Sampled EPLD and ASIC design projects with gate counts between 5 kgates and 25 kgates differ significantly with respect to total required effort (man-months) and design phase distribution. In the sample, EPLD design projects are completed with significantly less effort than ASIC design projects (EPLD: $\bar{x}=8.45$ and $s=6.34$ man-months; ASIC: $\bar{x}=19.24$ and $s=18.94$ man-months). Furthermore, ASIC designers spent significantly more relative effort on design verification. This result is in support of proposition 2.
19	Sampled EPLD and ASIC design projects with gate counts between 5 kgates and 25 kgates differ significantly with respect to the number of prototype iterations used until design completion (EPLD: $\bar{x}=13.90$ and $s=14.77$; ASIC: $\bar{x}=1.49$ and $s=1.48$). This result is in support of hypothesis 4.

Table VI.25b. Summary of results 11-19 (continued from table VI.25a).

Chapter VII

Summary and Conclusions

The theoretical and empirical results that I have presented in this thesis have explored the [changing] economics of experimentation and its impact on the design of new products and processes. In section VII.1 of this final chapter, I will briefly summarize the results presented in this thesis and then, in section VII.2, I discuss the implications for managerial practice and innovation theory. Suggestions for future research will be discussed in section VII.3.

VII.1 Summary of Results

In **chapter I**, I explained that the economics of experimentation are being radically affected in many fields due to the use of new and greatly improved versions of methods such as computer simulation, mass screening,

and rapid prototyping ("experimentation modes"). To illustrate these dramatic changes, I described two examples of fields (the design of integrated-circuit based systems and the design of pharmaceutical drugs) that are currently affected. Furthermore, I proposed that experimentation can be conducted in different "modes" (e.g. computer simulation, prototype testing) which dynamically vary with respect to the efficiency of repeated experimentation. Since repeated experimentation is a major innovation activity and accounts for a significant part of total design cost, I hypothesized that, as a design progresses, users will find it economical to switch between these modes so as to improve total design performance.

In **chapter II**, I reviewed three areas of the literature that are related to the research of iterative experimentation in design: (1) the study of experimental trial and error in problem-solving; (2) the study and modeling of iterations in design; and (3) the study of learning processes in design and manufacturing. Although some of the literature has helped me in my research, the literature review indicated that — to the best of my knowledge — no previous study has investigated and modeled the micro-level mechanism of experimentation from an economic perspective in general, or has investigated and modeled the economic consequences of new and improved experimentation modes for the design of new products and processes in particular.

In **chapter III**, I developed a general model of experimental trial and error that regards experimentation as an iterative four-step *design-build-run-analyze* cycle. For example, one might (1) conceive of and *design* a new, more rapidly-deploying airbag for a car; (2) *build* a prototype of key elements of that

airbag as well as any special apparatus needed to test its speed of deployment; (3) *run* the experiment to determine actual deployment speed; and (4) *analyze* the result. If the results of a first iteration are satisfactory, one stops. However, experimentation is usually a matter of repeated trial and error. That is, if analysis shows that the quality of a design can be improved cost-effectively, one modifies one's design during additional experimental cycles on the basis of what one has "learned" during previous cycles.

In the development of the general model, I discussed the content of each experimental step with respect to the elimination of design errors (which was studied in this thesis) and parameter design. In order to evaluate the efficiency of experimentation, three measures were defined: (a) the *cost* of conducting the four experimental steps; (b) the *accuracy* of the experimentation models used (defined as the probability of identifying a quality improvement opportunity if it exists and is tested for); and (c) the financial *quality loss* (i.e. the cost of not identifying an improvement opportunity if it exists).

Using the experimentation model, I proposed the following:

(1) The efficiency of experimental cycles varies with respect to conducting the four experimental steps within an experimental cycle.

(2) The efficiencies of repeated experimental cycles in selected modes (e.g. simulation, prototyping) decrease at different rates.

(3) As a result of (1) and (2), I proposed that, as a design progresses and a mode's efficiency decreases, there may be a point where switching to another mode is an economic strategy. This point was referred to as the "optimal switching point" (OSP) and I proposed that designers would

find it economical to revise their mode switching strategies as to move towards this OSP.

Furthermore, I suggested that changes in experimentation model accuracy and/or model build cost would shift the OSP, and designers will find that adjusting their mode switching strategies in the direction of the new OSP will result in significant reductions of total design cost.

In **chapter IV**, I developed a decision model that aids in the evaluation and selection of experimentation modes prior the start of an experimental cycle. As designers base their selection of experimentation modes on economic considerations, but assess them only intuitively and informally, the model can significantly improve the accuracy of their evaluations of mode efficiencies and ultimately reduce the cost of their experimentation activities.

In **chapter V**, I reported on my first field study on experimentation in custom circuit design during which I conducted approximately 30 detailed, face-to-face interviews with very experienced circuit designers over a period of three months. The information that was gathered during these interviews allowed me to achieve the following: (1) to compare simulation and prototype testing in the context of the general experimentation model that was developed in chapter III; (2) to examine behavior related to the mode switching from simulation and prototyping, and use the findings to refine the decision model that was developed in chapter IV; (3) to apply the decision model to a decision scenario encountered in the field study; and (4) to identify measures that can be used in studying and testing mode switching behavior in the large-scale field study of chapter VI.

In **chapter VI**, I reported on my second field study on experimentation in custom circuit design during which I mailed a 59-question survey to 1000 circuit designers (500 EPLD designers and 500 ASIC designers), resulting in a 46.7% response rate (39.7% of the responses could be used for data analysis). The survey provided me with a rich data set on the micro-level mechanism of experimentation in circuit design in general, and also enabled me to compare and contrast computer simulation and prototype testing in EPLD- and ASIC-based designs in particular. Some of the general findings of this field study are:

— An analysis of information that is available to designers prior to the start of experimentation shows that the efficiencies of simulation and prototype testing vary with respect to experimental steps (design, build, run, analyze) and that simulation is usually more effective at the start of experimentation.

— An analysis of information that is available to designers after the start of experimentation shows that the efficiency of simulation decreases with respect to repeated experimental cycles (and the fact that EPLD designers switch "early" suggests that the efficiency of prototype testing decreases slower than that of simulation).

— The data supported the proposition that designers who actively engage in mode switching can reduce total design cost significantly. This was in part supported by (1) the findings that EPLD designers tend to switch to prototype testing much earlier than ASIC designers do (i.e. the former actively engage in mode switching whereas the latter do not); and (2) a comparison of design projects of similar complexity showed that EPLD-

based designs outperformed ASIC-based designs by an effort factor of 2.3 (effort was measured in man-months).

— In addition to "learning by interference finding" (which was discussed in the literature review in chapter II), the data was also in support of a second learning process that I described as "learning about mode efficiencies". Due to the difficulty to accurately predict the path of problem-solving and the related mode efficiency trajectory prior to starting experimentation, designers iteratively learn about the changing mode efficiencies by paying attention to dynamic measures such as a simulation tool's "error detection rate". The results of what one has "learned about the efficiency of simulation" are then used to continuously rethink the decision of when to stop simulating and to start prototype testing (i.e. the switching point).

Taken together, these results appear to me interesting in several ways. First, they demonstrate that experimental trial and error and the related learning play a very important role in the determination of total design cost. Second, my data supports the hypothesis that design performance can be improved with the aid of an efficient mode switching strategy that has to be continuously adjusted and actively managed — particularly in the face of new and rapidly changing experimentation modes. Sometimes such an experimentation strategy may result in "getting it [the prototype] *wrong* the first time"; a strategy that may be counterintuitive from the perspective of prior research in product development. Third, it appears that learning rates (which affect both, "learning by interference finding" and "learning about

mode efficiencies") can be "accelerated" by the effective use of experimentation modes which raises the question if there are limitations on mode-assisted learning, or if learning can be accelerated indefinitely. More empirical work in this area is needed to explore this observation in greater detail.

VII.2 Implications for Managerial Practice and Theory

Mode Switching Strategy as a Novel Management Concept

The concept of explicitly managing the switching between modes of experimentation is — to the best of my knowledge — novel to the product development and innovation literature. In this thesis, I have shown that determining the point where it is optimal to switch and adjusting the mode switching strategy accordingly, can result in significant improvements of design performance. (To aid in this process, I have also developed a decision-support model that was developed as part of this thesis). Thus, managers will find it economical to provide design resources that aid in the determination of mode switching points and to devise managerial actions as to move designers' switching behavior as close as possible to the optimal point.

Implications for the Adoption of New Experimentation Techniques

Innovation practitioners and managers are now adopting and using new experimentation techniques without an explicit understanding of the change in experimentation economics that can follow, and the consequent change in economic design strategies that may result. One objective of this thesis is to help managers in understanding the impact of new and improved

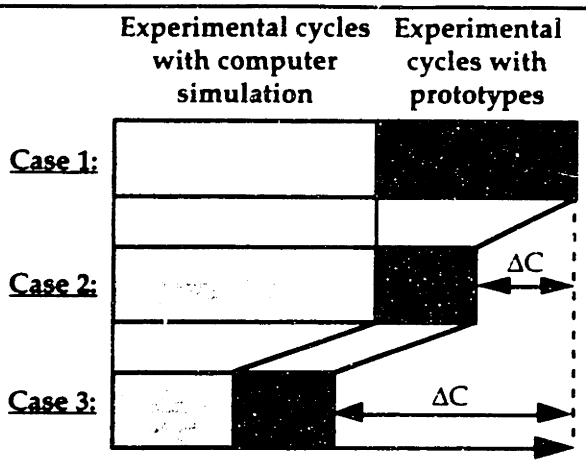
experimentation techniques on design performance and in formulating managerial actions that would motivate designers to take full advantage of these changes in experimentation economics.

To demonstrate the importance of correct managerial actions with a general example, consider the effect of rapid prototyping on a hypothetical design process XYZ[1] that consists of two phases: (1) experimental cycles with computer simulation; and (2) experimental cycles with prototypes (figure VII.1 on the next page). Before the introduction of the rapid prototyping technology, XYZ faces very high costs of building and modifying design prototypes. Thus an economical experimentation strategy includes the simulation of a design until one can conclude with a high degree of certainty that the first prototype will operate without failure (see case 1 in figure VII.1). The appropriate managerial action in this first case would include a development schedule with ample simulation time and resources, and perhaps performance feedback such as a "prototype-works-right-the-first-time" design award.

Now consider the availability of rapid prototyping which allows a designer to build (and modify) a prototype at a lower cost (see case 2 in figure VII.1). The use of rapid prototyping would most certainly reduce the total experimentation cost required for prototype testing, even if the managerial actions from case 1 were left in place and the mode switching strategy remained unchanged.

---

[1] The reader may notice that managers of XYZ faces similar issues that managers of ASIC design environments deal with after changing to an EPLD-based design approach.



Explanation:

Case 1: Before low cost prototyping technology is available.

Case 2: After technology is available but no modification of the mode switching strategy.

Case 3: After technology is available but with modification of the mode switching strategy.

Figure VII.1. The impact of rapid prototyping on design cost (with and without modifications of the mode switching strategy) (illustration).

However, the findings in this thesis suggest that additional opportunities for a significant design cost reduction would remain "untapped" unless management understands the impact of rapid prototyping on the mode switching point and revises its managerial actions accordingly. In this particular example, it would imply an "earlier" switching from simulation to prototype testing and, as a result, a change of the managerial actions that were appropriate for case 1. For example, the prior "prototype-works-right-the-first-time" design award would certainly motivate designers to follow experimentation practices that emphasize first-prototype success

(which was appropriate before the availability of rapid prototyping). Thus the award would have to be replaced by some incentive that would motivate designers to switch from simulation to prototype testing before being very certain that no errors remain in the design. This behavior, though somewhat counterintuitive, will often result in designers "getting it wrong the first time" but, at the same time, being able to further improve design performance significantly. (This final strategy of switching "early" is depicted in case 3 of figure VII.1.)

Implications for the Organization of Design Projects

Effective product and process design requires both that all of the organizational groups involved develop the appropriate specialized capabilities, and that the efforts of all these groups be integrated (Hayes, Wheelright, and Clark 1988). In the context of switching between experimentation modes, we sometimes find that the switching involves a hand-off between organizational groups that are specialists in the use of a particular experimentation mode. Such a hand-off is likely to occur with some "bias" unless the interaction and integration of these groups is managed effectively at both, the individual and organizational level.

Consider the use of computer simulation and prototype testing in integrated circuit design once again. In large design projects, one sometimes finds that simulation and prototype testing is assigned to different design groups, and that the knowledge and skills of these groups is highly specialized. Thus switching between experimentation modes will require a

formal hand-off between these two groups but may involve hand-off "biases". Some of the reasons (and possible solutions) are:

— Design groups that specialize in simulation know very little about [the economics of] prototype testing (and vice versa) and therefore are ill-positioned to evaluate the optimal mode switching (or hand-off) point. Thus, managers will find it beneficial (1) to continuously collect data related to the efficiency of simulation and prototype testing (e.g. by using electronic error elimination protocols); and (2) to periodically evaluate the decision to switch between simulation and prototype testing with the aid of an interdisciplinary group that consists of engineers from both groups.

— Design resources (e.g. time, staffing) are often allocated to groups prior to starting a project. However, I have demonstrated in this thesis that one cannot reasonably predict the optimal switching point (and, as a result, the total time required) prior to the start of simulation but instead has to rely on experimental "learning by doing" that occurs after the start of simulation. Unless experimentation-related design resources are dynamically reallocated (based on the learning that occurs during experimentation), it is unlikely that a hand-off from simulation to prototype testing (and vice versa) occurs at the optimal point[2].

Other reasons that may introduce hand-off "biases" relate to intergroup competition (e.g. the simulation group tries to demonstrate the goodness of

---

[2] For example, my second field study (chapter VI) showed that designers sometimes stop simulation because their preallocated "simulation time" has expired and that they are required to proceed to prototype testing.

simulation by handing-off an error-free — but excessively simulated — design to prototype testing). This list of reasons is by no means exhaustive but shows some of the organizational challenges that managers should consider in achieving effective product and process design.

Implications for Organizational Learning and Problem-Solving

To the extent that organizational learning is the sum of all micro-level learning processes in an organization, I have identified (and empirically verified) at least one more process which is built around the view that there are some things that designers can know in advance -- such as the general types of errors to be encountered (e.g. logic, timing, and signal quality) — but other matters can only be determined during the actual work of error detection (such as changes in mode efficiency which is contingent upon the number and distribution of errors detected during experimentation).

This view can be generalized by noting that there are limits on one's ability to predict the path of problem-solving, and therefore to selecting optimal modes and strategies for a course of experimentation in advance of actually doing the work. However, one can make some selections with respect to an experimentation strategy in advance, and can consciously make others during the actual course of the experimental work. Experimentation in particular, and trial and error procedures in general guarantee a problem solution only in the instance of "well-structured" problems — which are defined as those for which one can precisely specify a process of trial and error that will lead to a desired solution in a practical amount of time (Reitman 1965, Simon 1973, Pople 1982). For example, a traveling salesman problem

can be well-structured, because one can precisely specify a generator of alternate solutions and solution testing procedure that are guaranteed to eventually identify the best solution. However, "In general, the problems presented to problem solvers by the world are best regarded as ill-structured problems. They become well-structured only in the process of being prepared for the problem-solvers. It is not exaggerating much to say that there are no well-structured problems, only ill-structured problems that have been formalized for problem-solvers." (Simon 1973 p.186).

Ill-structured problems may involve an unknown "solution space" (a precisely specifiable domain(s) in which the solution is known to lie). They may also involve unknown or uncertain alternative solution pathways, inexact or unknown connections between means and ends and/or other difficulties. Ill-structured problems are solved by a process of first generating one or more (typically several) alternative solutions. These may or may not be the best possible solutions — one has no way of knowing. These alternatives are then tested against a whole array of requirements and constraints (Marples 1961, Simon 1981 p.149). Test outcomes are used to revise and refine the solutions under development, and progress is made in this way towards an acceptable result.

As an additional barrier to the predictability of an experimentation strategy, consider that problem statements are not necessarily stable as one conducts research or designs. An experimenter may be induced to change the object of a particular experimental search based on an unexpected experimental outcome. For example, a researcher looking for a new insulin-like drug might switch the topic of research to heart drugs if mass screening of

drugs for effects on blood sugar happens to reveal an effect that would be desirable in a heart drug. Further, changes in experimental goals in many field may be introduced by problem-solver who are "build into" the object or the context — as is the pilot of a plane or a machine operator or a manufacturing engineer in the case of a process machine.

Despite the just-mentioned barriers to predicting the course of experimental work in detail, my research suggested that one can learn to make experimentation strategies more effective by paying attention to what one can know that is useful to guiding such a strategy and when one can know it. Thus, computer companies specialize in repeatedly developing the next new computer model, and auto firms specialize repeatedly developing the next auto model. Under such conditions, one can learn useful things from prior projects that can help to guide the experimental work of the next project. For example, an engineer experienced in auto design work can look at specifications for a new model car and predict with reasonable accuracy the most difficult design problems that are likely to be encountered during the course of project.

In the instance of "very novel" projects, an ability to predict patterns of problem-solving (and the related experimentation strategy) at the outset of that project is equivalent to saying that we are able to predict the unexpected — not a very promising prospect. However, one can still make some predictions as the project unfolds. Consider what we know about the nature of the engineering problem-solving process. Marples (1961) and others have found that engineering problem definition and the related problem-solving can evolve as a project progresses. Thus, one may start a project only

knowing that one wants to build a computer that is "beyond the state of the art" with respect to speed at performing certain types of computation. But many designs may allow one to reach the specified goal, so the designer's first step is to generate and analyze alternative approaches in a preliminary way. (E.g.: "We could use a single superfast processor or multiple processors in parallel".) As work proceeds, the different subproblems that are associated with each approach become clearer. (For example, engineers may identify the key subproblems associated with the parallel processor approach as innovative computer architecture and software. In contrast, the key subproblems in the instance of a single processor approach may be determined to be the design and fabrication of a processor faster than any made before.) Since one knows something about the nature of each approaches prior to designing the related experimental work, one is able to use this knowledge to improve the likely efficiency of the likely unfolding of experimental work during the course of even a very novel project.

VII.3 Suggestions for Future Research

The results of this thesis open up a number of interesting and important research issues. The first is that we have seen the impact of new and dramatically improved experimentation modes on the design of custom circuits and I suggest that there are many other exciting fields to be researched that are undergoing similar changes. For example, in the first chapter of this thesis I have briefly discussed the economics of "mass screening" and "rational drug design", two rapidly-evolving experimentation modes used in the development of new pharmaceutical drugs. Advances in the former

mode are being driven by advances in combinatorial chemistry, which has radically reduced the costs of creating known molecules which are tested via mass screening. In contrast, advances in the latter are being driven by improvements in researchers' abilities to simulate molecular interactions with computers. Because of the effect of "mass-screening" on the economics of drug development, firms are likely to utilize both modes and switch between them as to economize on the difference in efficiencies at various points of the development process. As the fortunes of even large firms depend on the development of a few important drugs (and experimentation accounts for a significant part of the total development cost and time), one of the most important managerial decision variables will involve finding the optimal switching point(s) between "mass screening" and "rational design". Thus, research into determining the economic variables that drive mode switching in pharmaceutical drug development is not only interesting from a research perspective but will also have very significant commercial implications for pharmaceutical firms.

Another important research question is how the success of new and improved experimentation modes will affect the core competency of a corporation (Prahalad and Hamel 1990). Sometimes, the introduction of new modes requires new competencies and/or make previously necessary firm competencies superfluous. For example, the core competence required for mass screening in the design of pharmaceuticals is built upon a good understanding of a particular search technique, while the ability to competently perform rational drug design is built upon an understanding of molecular design. In a study of core competencies in ethical drug discovery,

Henderson (1993) found that the "old style" of drug discovery (i.e. large-scale screening of compounds in screens designed to mimic disease states in man, without a good understanding of the specific biochemical and molecular pathways that are responsible for a therapeutic effect) required a group of pharmacologists who design and run screens and a group of organic chemists who were responsible for synthesizing variants of promising compounds. In contrast, "new style" drug discovery (i.e. rational design) requires the integration of rapidly changing knowledge from scientists with training in many disciplines, ranging from molecular biology to synthetic chemistry. And with the recent dramatic advances in "mass screening", firms may find that some of the competencies required for rational drug design may prove to be superfluous. Thus, research into the impact of new experimentation modes on firm competence can provide new insights on how firms ought to adapt to these changes.

Finally, a very interesting research issue is the study of how experimental learning varies with the state of knowledge of a given technical or scientific field. For example, Pisano (1994) studied how an emphasis on learning by laboratory experimentation impacts development lead times in two different environments: traditional chemical-based pharmaceuticals and new biotechnology-based pharmaceuticals. He found that in the processing of chemical-based pharmaceuticals — an environment that is well-understood in theory and practice — more emphasis on laboratory experimentation before transferring the process to the factory is associated with faster development. In contrast, in the processing of biotechnology-based pharmaceuticals — an environment that is not well-understood in theory or

practice — more emphasis on laboratory experimentation does not seem to shorten development lead times. Similarly, in my study of error elimination in circuit design (chapters V and VI), I have found that for some error classes (e.g. signal quality errors), the complex causal pathways leading to a design failure are poorly understood and very difficult to simulate. Thus, the rate of experimental learning in simulation is significantly limited if compared to learning in the testing of physical prototypes. This opens up interesting research issues in the field of organizational learning in general, or the study of limitations in learning rates in particular. (For example, what are the limiting [economic] factors of learning rates in organizations and, as a consequence, how can learning be accelerated?)[3]

The list of implications and suggestions is by no means exhaustive, but the discussion in this chapter shows that more research into the impact of new and dramatically improved experimentation modes and the related experimental learning strategies is needed. I hope that this thesis serves as a starting point of a new stream of research into these areas, and that the insights and decision tools that follow from my work have a significant impact on managerial practice and theory ■

[3] As a very general example, consider the recent emergence of managerial "microworlds" as a learning laboratory for managers (Senge and Lennon 1990). "Microworlds" is a term coined by Seymour Papert at MIT and describes an interactive computerized environment that simulates a real-world situation. In these simulated microworlds, decision makers take control of a fictitious company or business situation that they normally fill in real life. But in contrast to the real world, managers can experiment with business strategies and policies without jeopardizing the future of their company. Since a micro-world tries to simulate the complex interactions of firm with its markets and competitors, managers are free to experiment and learn about long-term systematic consequences of their actions at a much lower cost than in the real-world.

Appendix A

Mathematical Derivations

Error Detection and Effort

In the following simple derivation, I would like to show that the effort (using time as a proxy) required in error detection increases as more errors are detected.

Let:

$E(t)$ = Cumulative errors detected at t (normalized to 1)
$e(t)$ = Residual errors at t (normalized to 1)
$p(t)$ = Rate of error detection at t (normalized to 1)

Assume:

The rate of error detection is proportional to the pool of residual errors:

$$\Rightarrow \quad p(t) \propto e(t)$$
$$\Rightarrow \quad p(t) = \gamma e(t)$$

Then:

$$\Rightarrow \quad \frac{\partial E(t)}{\partial t} = p(t) = \gamma e(t) = \gamma(1 - E(t)) = \gamma - \gamma E(t)$$
$$\Rightarrow \quad \frac{\partial E(t)}{\partial t} + \gamma E(t) = \gamma$$

Solve differential equation:

$\Rightarrow \quad E_1(t) = A \times e^{-t}$ *(homogenous solution by inspection)*

$\Rightarrow \quad E_1(t) = 1$ *(particular solution by inspection)*

$\Rightarrow \quad E(t) = 1 + A \times e^{-t} \therefore$

Substitute initial conditions:

$\Rightarrow \quad E(0) = 0$

$\Rightarrow \quad A = -1$

$\Rightarrow \quad E(t) = 1 - e^{-t} \therefore$

We finally arrive at an exponential function that follows the shape depicted in figure A.1. Thus we can see that the error detection rate decreases exponentially with time (and cumulative errors detected). In other words, detecting an error will become increasingly time-consuming as more errors are detected.

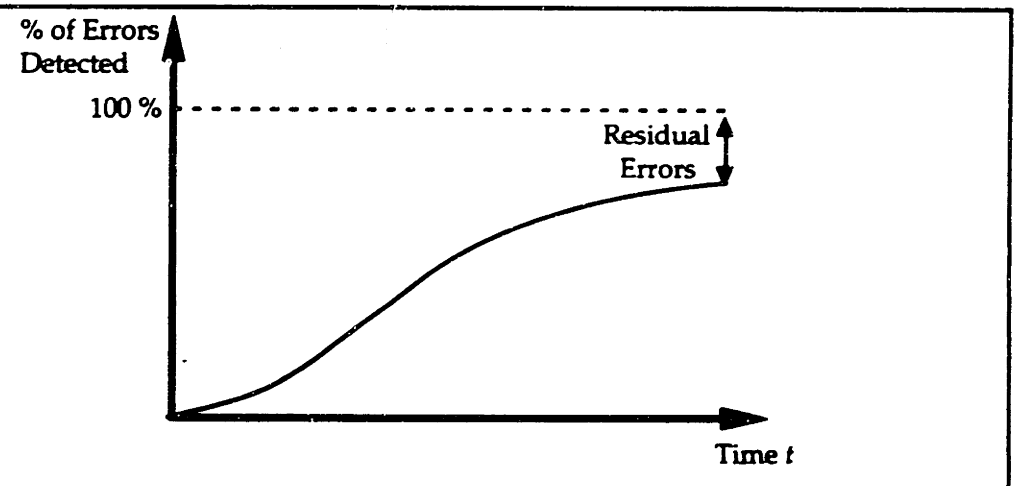


Figure A.1. Cumulative errors detected as a function of time.

Please note that the discussion has so far assumed that the detection mode is perfectly accurate (i.e. all errors will eventually detected as $t \rightarrow \infty$). If we include a term α (where $0 \leq \alpha \leq 1$) for model accuracy, one can expect to find a residual error term of $\alpha \times 100\%$ in $E(t)$ as $t \rightarrow \infty$.

Appendix B

Notes on the Design of Custom Circuits

B.1 Introduction

My field studies were on experimentation strategies in the design of custom circuits that contain electrically-programmable logic devices (EPLDs) and application-specific integrated circuits (ASICs). This appendix is written for the reader who is mostly unfamiliar with these two technologies and therefore would like to receive additional background material. Some of the material may not be directly relevant to the reported field studies on experimentation but it nonetheless provides the reader with an opportunity to improve his/her understanding and appreciation of my empirical findings.

I divided the background material in the following three sections (which can be read separately because they are fairly independent) :

— The first section (B.2) describes the different technologies that have been driving custom integrated circuit design and fabrication. In particular, I analyze the evolution of the customer/manufacturer interface which is very important for understanding the difference between design technologies.

— The second section (B.3) provides some market information that underscores the commercial significance of custom integrated circuits.

— The third section (B.4) provides a general overview of the circuit design process by taking the reader through the major steps in designing an ASIC and an EPLD.

B.2 Integrated Circuit Design Technologies

Integrated circuits (ICs) have revolutionized modern product development as they can be found in just about any system ranging from children's toys to modern airplanes. They consist of a large number of components, normally transistors, built into the surface of a silicon wafer. Modern integrated circuits, such as microprocessors, can easily exceed one million working components, operating at dimensions invisible to the human eye. The fabrication of modern ICs can consist of more than 500 individual manufacturing steps, each so complex that it requires a high level of expertise and manufacturing control. Since a single dust particle can disable the functionality of an entire IC, manufacturing environments have

to be ultraclean. Integrated circuit design and fabrication are without question some of the most complex processes today.

B.2.1 Full-Custom Integrated Circuits

One important branch of integrated circuit design is the development of custom applications. In contrast to standard ICs which are designed for a wide range of applications, they can be defined in the broadest sense as ICs designed for a particular application or end-use such as in a compact disc player or a telecommunication system. In designing custom integrated circuits, two sources of information are central to problem-solving: 1) information at the integrated circuit customer locus involving a rich and complex understanding of both the overall application in which the custom integrated circuit will play a role and the specific function required of that circuit; 2) information at the circuit manufacturer locus involving a rich understanding of the constraints and possibilities of the design and fabrication processes that the manufacturer uses to produce integrated circuits (von Hippel 1993) (see figure B.1).

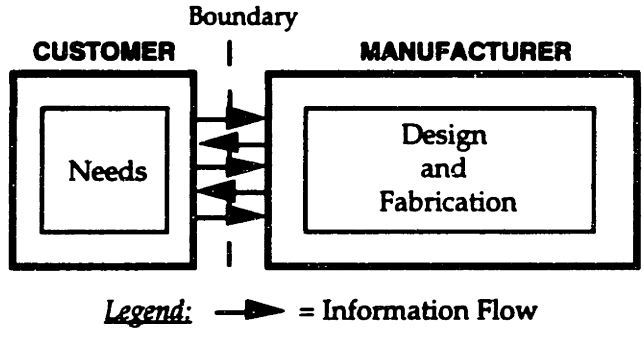


Figure B.1. The customer/manufacturer interface.

Traditionally, custom integrated circuits were developed in an iterative process between circuit customer possessing need-related information and an integrated circuit manufacturer possessing information about designing and fabricating integrated circuits. The process would begin with a customer specifying the functions that the chip was to perform to a circuit design specialist employed by the integrated circuit manufacturer. During the design much information transfer would be required between customer and manufacturer as to clarify complex requirements and/or to negotiate design alternatives. The integrated circuit would then be designed at the manufacturer locus, and an (expensive) prototype would be produced and sent to the customer. Testing by the user would typically reveal faults in the chip and/or initial specification, responsive changes would be made, a new prototype built, and so forth.

Because of high prototyping and information transfer costs, designing custom ICs was so expensive that it created a barrier to entry for potential customers. (Prototypes cost in excess of $100k since they typically involved a long process that started with the design and manufacture of numerous photolithographic mask sets, continued with many days or weeks of processing, and ended with elaborate tests — all on extremely expensive equipment with a useful life of 2-4 years). Instead, customers resorted to standard integrated circuits and designed their systems around these standard components. This resulted in designs with lower performance and lower degrees of integration.

B.2.2 Application-Specific Integrated Circuits (ASICs)

In 1981, a company named LSI started to offer customized IC design and fabrication services at a cost well below other traditional IC manufacturers (Walker 1992). The integrated circuits they offered were named ASICs, or application-specific integrated circuits. LSI was able to employ two significant innovations: (1) a reduction of custom fabrication costs; and (2) an increased utilization of computer-aided engineering tools in circuit design.

With respect to (1), fabrication was repartitioned into two tasks, customization and standard fabrication. During standard fabrication, silicon wafers with arrays of standard circuit elements ("sea of gates") are fabricated. These standard circuit elements arrays are designed by the manufacturer to be interconnectable into working integrated circuits by the later addition of custom interconnection layers ("the customization") designed in accordance with specific customer needs. Thus, LSI was able to reduce fabrication cost and time by manufacturing standard and nearly-completed gate array wafers, ready to be customized with 1-2 final mask layers.

With respect to (2), improved design tools allowed designers to simulate design performance prior to fabrication and therefore increased the probability of a first-time design success (which reduced overall prototyping cost and time).

Since ASIC technology allowed design and fabrication to be nearly independent with respect to problem-solving and ASIC design tools became more user-friendly, customers started to invest in the development of in-house design capability. They started to buy computers, design tools, and trained their own engineers in integrated circuit design. Thus an increasing

number of customers were developing designs in-house instead of resorting to external design specialists[1]. The overall problem of designing custom circuits is now partitioned into subproblems which draw on only one locus of information, thereby eliminating the need to iterate between customer and manufacturer sites in the design process. The manufacturer of ASICs draws on its own information to develop and improve the fabrication process in its manufacturing plant, a "silicon" foundry. And the customer draws on local need information to design and customize the IC. The transition from conventional IC design to ASIC design is shown in figure B.2.

While ASIC design has been a substantial improvement over traditional custom IC development, its fabrication can still be quite costly and time-consuming. Non-recurring engineering (NRE) costs for each prototype iteration can cost tens of thousands of dollars and may take several weeks[2]. (If changes to the prototype are minor, some specialist firms can make design modifications on-chip with aid of sophisticated ion-beam equipment — a service that can still cost several thousand dollars. Sometimes it is possible to solve a prototype error with a design workaround which will probably delay the project and absorb significant design resources.) As a result, there is much emphasis on first-time prototype success, as shown by company awards such as a "first silicon design prize". Since designers have much to lose from

---

[1] A significant number of ASIC design specialists still exist in the form of local design centers that support customers in all aspects of the design process, particularly those who have fewer incentives to invest in in-house design capability.

[2] Some ASIC foundries have been trying very hard to reduce prototype turnaround time to a few days. While there have been remarkable improvements over the last year, the fabrication of ASICs still has an intrinsic cycle time of a few days that is very difficult to reduce.

prototype iterations, they focus most of their resources on computer simulation.

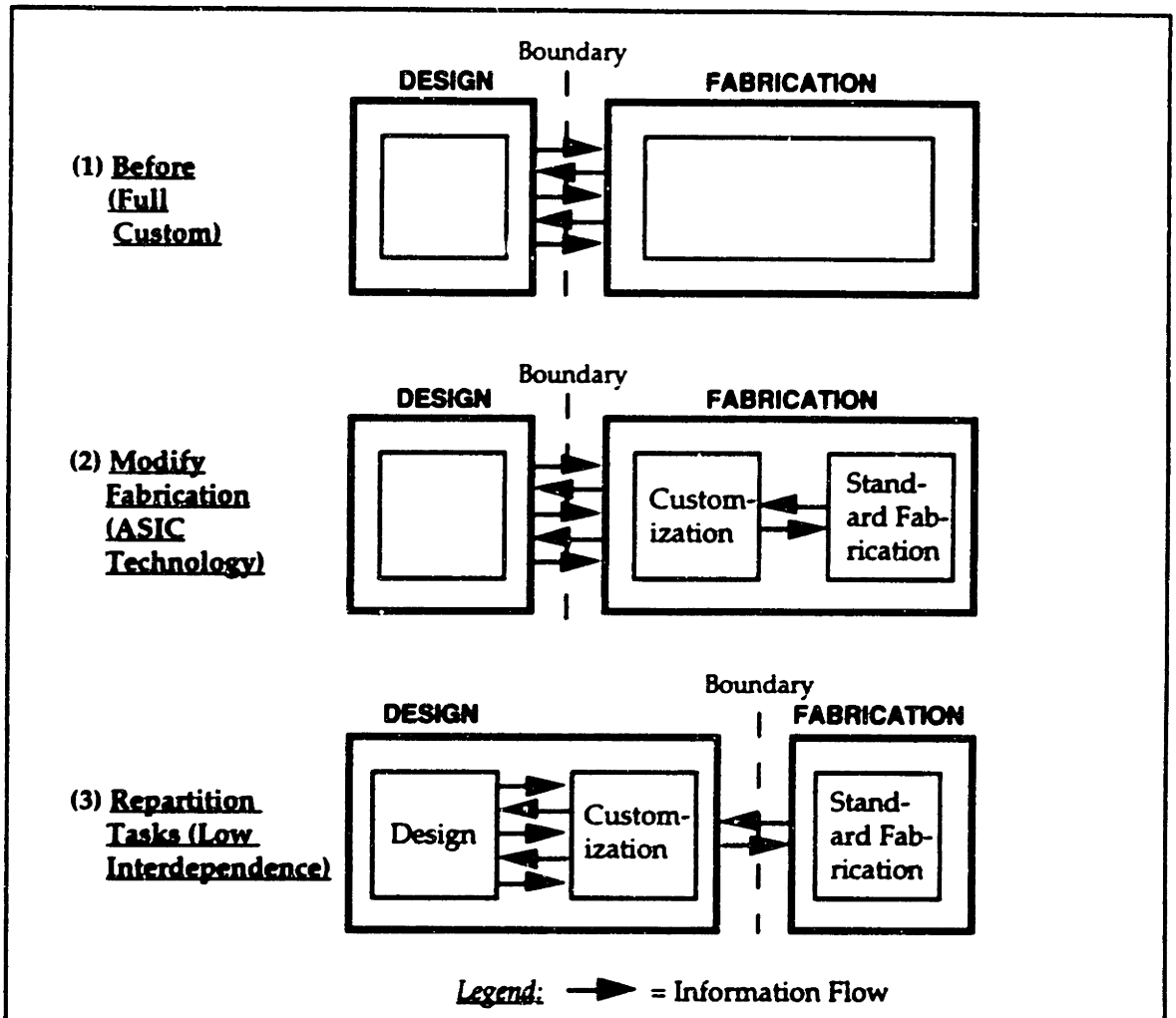


Figure B.2. In the traditional approach, design and fabrication are highly interdependent (with respect to problem-solving) and it was economical to keep both tasks within a firm. With the emergence of ASIC design, design and fabrication could be repartitioned as to minimize problem-solving interdependence. Thus it became economical to move design to the locus of need related information, i.e. the customer[3].

[3] This argument is based on the assumption that information transfer is costly, or sticky (see von Hippel 93). As cross-boundary information transfer is reduced, information transfer cost is internalized, and presumably occur at a much lower cost. This argument is analogous to a reduction of transaction cost economies in vertically integrated firms.

B.2.3 Electrically-Programmable Logic Devices (EPLDs)

During the second half of the 1980s, a new generation of electrically-programmable logic devices (EPLDs) significantly changed the field of custom circuit design. While very primitive EPLDs had been around for a long time[4], it was only after the introduction of higher density devices and low-cost and easy-to-use engineering design tools (software and hardware) that EPLDs rapidly gained in acceptance.

In contrast to ASICs where prototyping costs are relatively high (because of NRE costs) and prototype turnaround may take weeks, EPLD technology permits designers to modify prototype designs (1) at little cost; (2) in a short time; and (3) at the designer's locus. This was accomplished by inventing an IC technology that has flexible interconnects and consists of an array of blocks that can be configured to perform various logic functions, without requiring additional fabrication steps. These flexible interconnects can be routed via simple desktop programming devices, allowing user's to produce working prototypes at their workplace[5] and immediately test them in designated systems.

---

[4] For example, programmable array logic (PAL) had been available for many years but it suffered from low gate densities and poor computer-aided design tools.

[5] There are various technologies available for "desktop prototyping". For example, EPROMs can be electrically programmed and its memory remains nonvolatile. Reprogramming can be achieved by exposing the open circuit area (normally covered by a quartz window) to ultraviolet light. Or field-programmable gate arrays (FPGAs) are often made of small static memory elements that are used to make interconnections by controlling an N-channel transistor. These transistors can be activated or disactivated by applying an appropriate voltage to them. An overview of programming technologies can be found in Vopni (91) and is partially shown in the footnote section of the next page.

From a task-partitioning perspective, EPLD-based designs have low problem-solving and production interdependence between design and fabrication (see figure B.3).

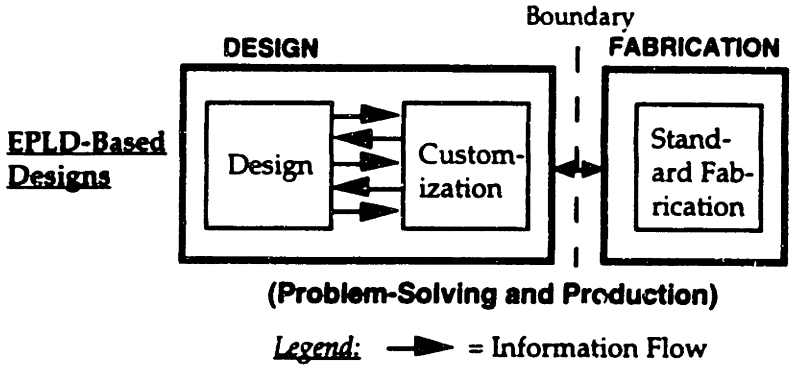
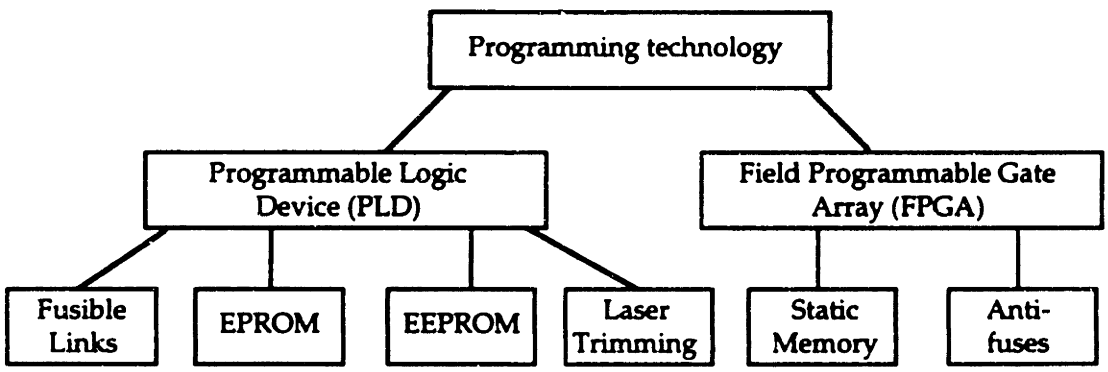


Figure B.3. EPLD-based designs have virtually no problem-solving and production interdependence between design and fabrication. (There is some negligible information transfer about general user needs with respect to non-dedicated EPLD architectures. The information is used to select programmable device families that are particularly suitable for a specific design.)

The manufacturer focuses mainly on advancing device technology (i.e. increasing density and speed while reducing fabrication cost) but has little knowledge about specific designs realized at the user site. Furthermore, EPLD design tools are typically lower in cost, easier to learn but offer fewer



simulation capabilities, when compared to ASIC design tools. We will later see that this difference can be partially explained by lower incentives to invest in EPLD simulation tools because of a lower cost to modify prototypes (and, as a result, lower economic benefits from simulation).

In terms of technological limitations, it should be pointed out that currently available EPLDs do not allow densities above 25,000 gates per device and thus several EPLDs have to be combined for larger designs[6]. (In contrast, ASIC designs can exceed 100,000 gates.)

Finally, there is also a difference in unit cost per device which is especially critical to designs that will be manufactured in large volume. In the case of EPLDs, the fixed cost is small since the devices are bought off-the-shelf and programmed in-house. Thus there is little change in the average unit cost as the production volume increases (other than standard volume discounts). On the other hand, ASICs have a fairly large fixed cost component (NRE costs) which can be amortized over increases in production volume and the variable unit cost is significantly lower than for EPLDs. As a consequence, designers sometimes calculate a crossover point (in units manufactured) where both technologies result in similar total production cost (see figure B.4). The more effective design technology (i.e. EPLD or ASIC) is selected as a function of the expected units to be manufactured.

However, recent developments have rendered the analysis in figure B.4 as insufficient. First, an increasing number of specialist firms can convert

---

[6] In my second field study (chapter VI), I found that 151 out of 174 EPLD-based hardware designs reported (86.8%) had densities at or below 25,000 gates. In contrast, 69 out of 171 ASIC designs reported (40.4%) had densities at or below 25,000 gates. I suspect that many of these ASIC designs could have been implemented with EPLDs.

EPLD-based design directly to ASICs in order to take advantage of lower production unit costs. While this option may not be feasible for all EPLD-based designs today, many of the technical obstacles in converting EPLDs to ASICs have been and continue to be solved. Second, the price/performance ratio of EPLDs is falling very quickly as EPLD vendors are heavily investing in the development of faster and denser programmable devices[7].

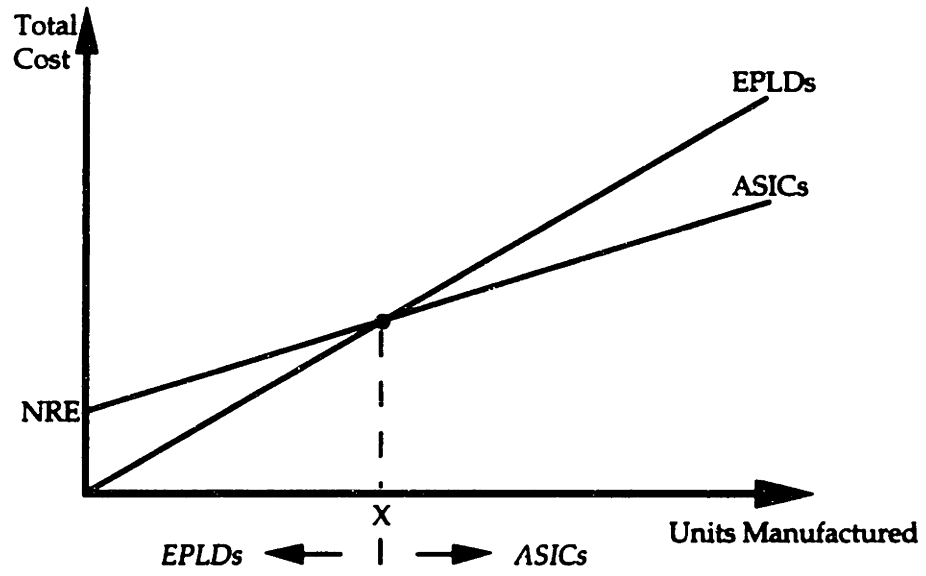


Figure B.4. The fixed cost component of ASICs can be amortized over multiple units and therefore its average cost decreases with volume. In contrast, EPLD-based designs are driven largely by variable cost and their average cost changes little with volume. As a result, there is a critical volume X (where it makes sense to switch) that customers often calculate before choosing the appropriate design technology.

[7] Third, my empirical findings in chapter VI suggest that EPLD-based designs are developed with significantly less effort than ASIC designs.

B.3 The Commercial Significance of Custom ICs

The custom integrated circuit market is valued at about $11.4 billion (1993) which accounts approximately for 16% of the total world-wide semiconductor market of $70.2 billion (see table B.1 on next page). A 16% share is a significant size if one considers the relatively recent emergence of custom IC technology.

Within the custom IC market, gate-array ("sea of gates") and standard cell technologies are the dominant market drivers. In contrast to gate arrays which I have explained earlier, standard cell technologies use standard macro-cell libraries in the design and fabrication of new circuits. As a result, they require complete mask sets for all layers of fabrication which drives the prototyping cost substantially higher than for gate arrays (the prototyping cost is comparable to full-custom fabrication). The advantage of standard cells, however, is (1) the use of predefined macro-cells during design which speeds up the overall design process; and (2) the achievement of higher design densities relative to gate array technology (but lower densities relative to full-custom).

The most interesting pattern in the market size and growth data (table B.1) is the relatively rapid growth of EPLD technology. The EPLD market is expected to grow at a compounded average growth rate (CAGR) of 19% between 1991 and 1998. Even after discounting the difficulty of predicting market growth, such a figure makes EPLDs the fastest growing segment within the custom integrated circuit market.

SEGMENT	1991	1992	1993	1994	1995	1996	1997	1998	CAGR %
MOS Gate Arrays	2,845	2,915	3,515	3,675	3,950	4,350	4,700	5,100	9
Bip. Gate Arrays	1,000	910	785	750	675	615	540	460	-11
Linear Arrays	210	240	260	280	300	330	350	375	9
Σ Semicustom ASIC	4,055	4,065	4,560	4,705	4,925	5,295	5,590	5,935	6
Standard Cell	2,120	2,290	2,920	3,375	3,850	4,600	5,350	6,200	17
Full Custom	2,625	2,650	2,700	2,725	2,750	2,800	2,850	2,900	1
Σ Custom ASIC	4,745	4,940	5,620	6,100	6,600	7,400	8,200	9,100	10
Bipolar PLDs	335	280	230	185	150	120	95	75	-19
FPGAs	170	245	365	500	650	900	1,100	1,350	34
Other MOS PLDs	400	455	645	785	910	1,150	1,325	1,550	21
Σ EPLDs	905	980	1,240	1,470	1,710	2,170	2,520	2,975	19
Σ ASIC & EPLD	9,705	9,985	11,420	12,275	13,235	14,865	16,310	18,010	9
ASIC & EPLD adj.*	7,080	7,335	8,720	9,550	10,485	12,065	13,460	15,110	11
Percent Change	7	3	14	7	8	12	10	10	-
Percent Change*	10	4	19	10	10	15	12	12	-
Percent Change Total IC Market	9	12	30	8	13	18	12	10	-
Worldwide IC Mkt	48,265	53,970	70,150	76,020	86,000	101,500	113,700	125,000	15
ASIC & EPLD % of Worldwide IC Mkt	20	19	16	16	15	15	14	14	-
ASIC & EPLD* % of Worldwide IC Mkt	15	14	12	13	12	12	12	12	-

* *Not including full-custom.*

Table B.1. ASIC & EPLD Market Forecast. Source: Integrated Circuit Engineering (ICE) Corporation.

B.4 The Custom IC Design Process

In order to illustrate the process of designing custom integrated circuits, I have constructed a flow chart consisting of the major steps in the design of complex ASICs[8] (see figures B.5a and B.5b on the next two pages) and they will be discussed in this section.

ASIC Design

The process starts with the general formulation of design requirements (diagnostic testability, microcode, board-level, and product planning) and is directly followed by chip-level decisions such as estimates of pin counts, design size, etc. These early phases of the design can be best characterized as "design specification" and often involve meetings with engineers from the ASIC foundry ("the vendor"). During these meetings, technical issues that relate to the interdependence between design and fabrication are discussed, technical and economic trade-offs are considered, and contractual agreements between designer ("the customer") and vendor are made[9]. After an agreement is signed, the vendor usually supplies software design libraries that are

---

[8] The information shown in the design flow is based on a combination of field interviews and publications on ASIC design.

[9] Some of the terms that can be found in such a contract are:
- The amount of non-recurring engineering charges (NRE) the designer will have to pay for in the manufacture, test, and delivery of prototypes and production devices.
- The tasks to be performed by the designer and the items to be delivered (e.g. final design specifications, test vector sets, etc.).
- The tasks to be performed by the foundry and the items to be delivered within a specified time window (e.g. layout, back annotation, etc.).
- Cancellation and rescheduling provisions if design modifications occur.
- Service options and pricing.

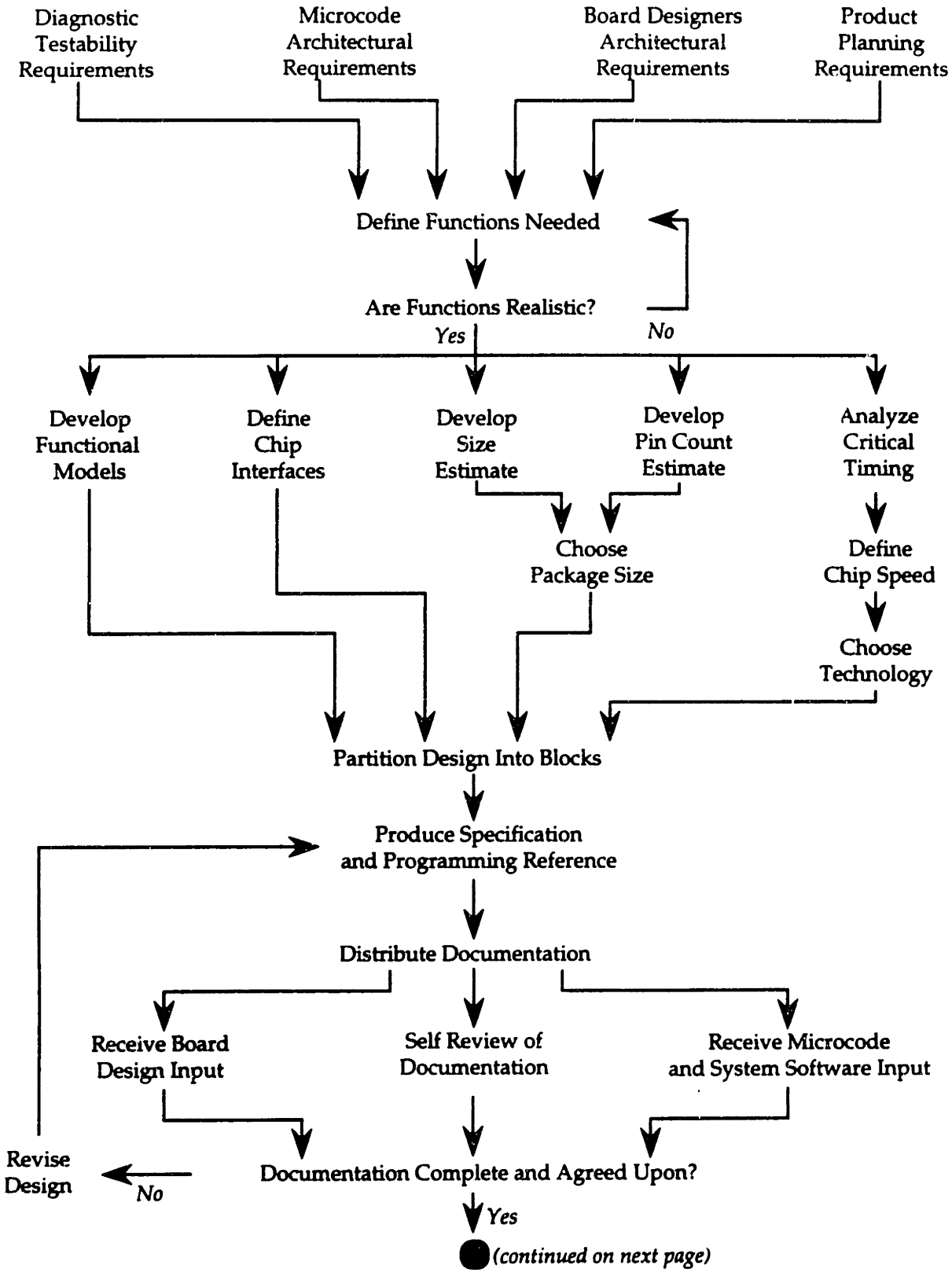


Figure B.5a. ASIC design flow (continued on figure B.5b).

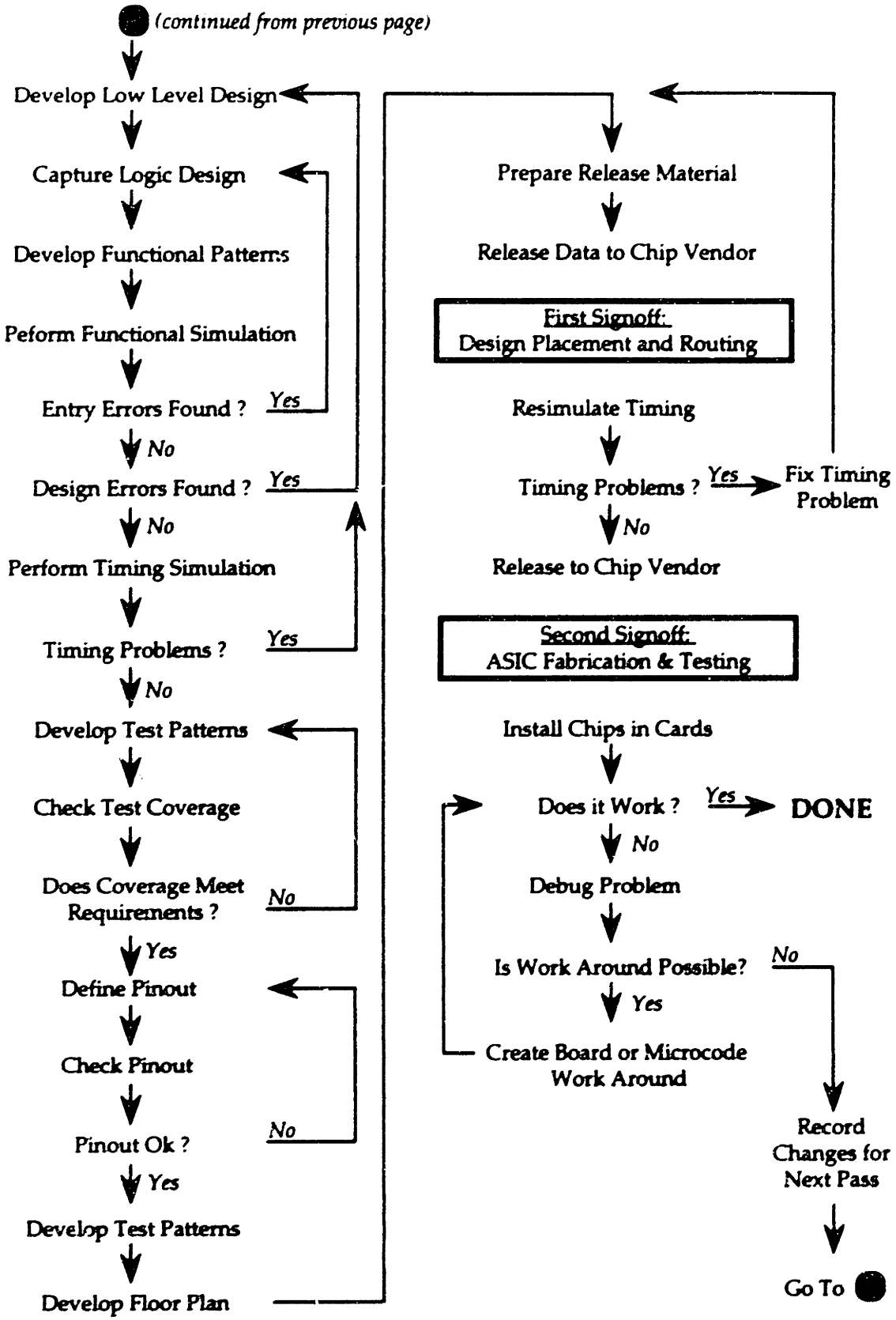


Figure B.5b. ASIC design flow (continued from figure B.5a).

specific to its ASIC technology. These libraries are needed to start the low-level design. Furthermore, in very complex projects, designers often write a completed set of design documentation before starting the actual design development.

Next, designers develop the low-level design according to specifications and with the aid of either schematic entry tools ("schematic capture") or higher level design synthesis tools. This phase can be characterized as "design development".

After a design block is completed to the extent that it can be simulated, the designer develops a set of functional patterns (or functional vectors) and runs a functional simulation. If an error is detected, it is analyzed for its cause and design changes are made. This process iterates until no functional errors are detected. Functional simulation is followed by timing simulation and again designers iterate until no error is detected. Since the ASIC has not been layed out or routed, timing delay models are only rough approximations and are of moderate accuracy. As a consequence, some timing simulation has to repeated when the routed design is available (later in the design process). The simulation phase can be described as "design verification".

After the design is verified with the aid of computer simulation, the designers have to prepare a design package for the ASIC vendor. Such a design package normally includes (1) a set of test patterns used by the vendor to verify the fabricated ASIC for physical functionality; (2) the translated design files (in a standard interchange format); (3) copies of error reports; and (4) a copy of the design floorplan. Upon receipt of the design package, the ASIC vendor reviews the material and prepares a design specification

document that has to be approved and signed by the customer. The handover of a signed specification document is often referred to as the "first sign-off". After receiving the signed document, the vendor develops a design layout and routes the array interconnects (in the case of gate-arrays) which are highly dependent on the fabrication process. The actual lengths of metal interconnects are extracted and used to create an interconnect-delay file which is returned to the customer. In addition, the vendor normally runs its own simulations to verify the manufacturability of the design. Using the interconnect-delay file, the customer can rerun timing simulations with higher modeling accuracy and check for errors that passed earlier timing simulations. If errors are found, the design has to be modified and part of the design process has to be repeated. Once the customer is satisfied with the design, it is formally released to the vendor by signing a release-to-manufacturing approval (often referred to as the "second sign-off"). The process of re-simulating with more accurate models can still be regarded as "design verification" (or better "design re-verification").

After receiving the signed release approval, the vendor generates the final design database which is used to create pattern-generation (PG) tapes. The PG tapes are used to manufacture photolithography masks which contain information on the IC customization layer(s). Then the integrated circuit is fabricated (for a detailed description of the actual fabrication process, see Einspruch (1991)). The first prototypes are tested and packaged by the vendor and delivered to the customer for in-system verification. The steps involved to create a prototype can be described as "design prototyping" (from the

customer perspective, however, "design prototyping" may only involve the time spent on prototyping-related issues with the vendor).

Finally, the designer determines if the prototype works without error which is not always the case ("design evaluation"). If an error is detected, costly prototype modifications are necessary unless a hardware or microcode workaround can be found. If a workaround is not possible, the design has to be modified and most of the earlier design and sign-off related steps have to be repeated. After the prototype performs as intended, a prototype approval is signed and volume production can begin. Due to the high cost of correcting design errors, ASIC designers rely heavily on simulation to get the prototype "right the first time".

EPLD Design

The design of EPLDs follows a similar process but due to the ability to prototype ICs in-house, there are a number of differences compared to the process of ASIC design:

— Less effort is required in the initial specification phase due to the lower cost of incorporating [physical] design changes later in the design process.

— Usually, EPLD designs do not require meetings or contractual agreements (e.g. no sign-off) with vendors since the design is not handed over to an outside party.

— EPLD design software can route a design directly after it is completed and thus the most accurate timing delay estimates available can be used at

the start of timing simulation (i.e. no re-simulation with more accurate models is necessary).

— EPLD designers do not have to prepare test vectors that ensure the physical functionality of a prototype device since EPLD manufacturers test the devices before delivering them to the customer.

— While the first hardware board can be expensive, prototype modifications are normally much less costly than for ASICs.

— Because prototype modifications are less costly, EPLD designers depend less on computer simulation than ASIC designers do. In fact, they may find it more effective to conduct simulation and iterative prototype testing which may result in getting their prototypes "wrong the first time". (This observation will be discussed in other parts of this thesis.)

Appendix C

First Field Study: Notes on Design Errors

Notes on Design Errors Detected in Prototype Testing

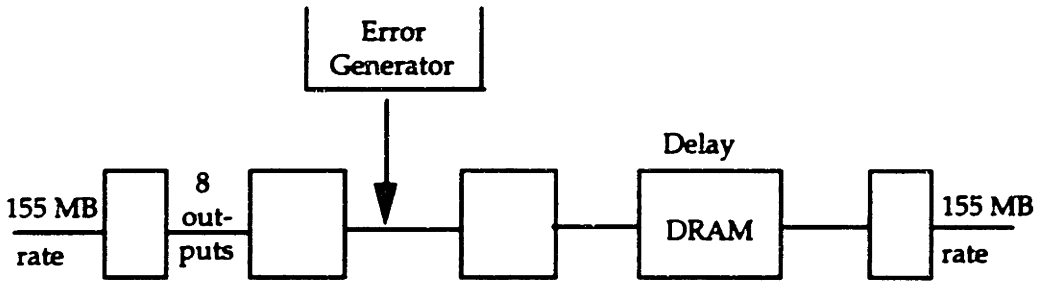
On the following pages, I will provide notes on errors that were first detected in prototype testing (i.e. *after* switching from simulation to the first prototype). The notes were taken during interviews with hardware designers that predominantly use EPLDs in their designs (first field study). Since the notes are intended as additional reference material only, I left them as they were written during the interview (i.e. they are not completely self-explanatory). The code letters AK, LD, PC, and RP stand for the designers interviewed and the error number is always in reference to the interviewee. Thus the code AK-1 stands for the first error reported by designer AK. For a description of the data gathering process, please consult chapter V.

Error Report Note #1

Error Code: AK-01
Error Class: Logic
Interview Date: March 18, 1994

1. Overall design problem

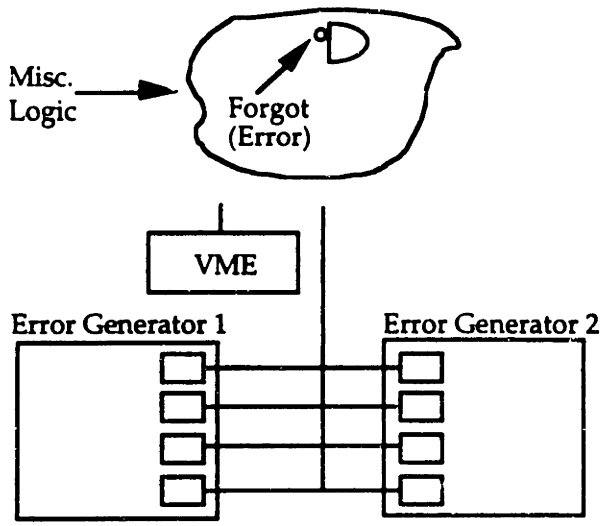
Error generation chip for an emulation of fiber optic links across the country (incl. dual VME and dual error generator). Customer wanted to check data communications protocol and needed an error generation capability that helped simulating errors in the field. The alternative would have been to rent a genuine fiber optic cable and test it for a year (expensive).



The project is internally known as the Long Link Emulator.

2. Description of error

Missed a bubble (negation) on one of the logic gates; the bus didn't work (read VME location, received bus time out).



3. Description of error cause

AK simulated the VME and both error generators. After the simulation, he had to change some things on the VME (names of pins), but somehow missed the bubble. He did not simulate after the change and therefore the error went undetected.

4. Method of error detection and analysis

Someone else ran some tests on the prototype and noticed a problem (bus didn't work). With the help of a logic analyzer and some resimulation, the problem was found.

5. Reason for not detecting error during simulation

Didn't resimulate after change was made.

6. Method of error correction

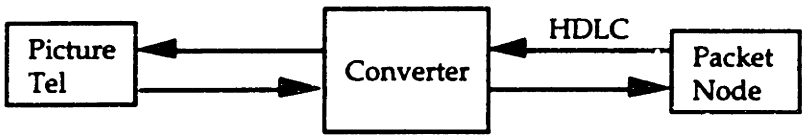
Added bubble to design; reprogrammmed EPLD. Chip had only 50% logic utilization and therefore it was easy to add logic. (Removing chip from board took about 1/2 day.)

Error Report Note #2

Error Code: AK-02
Error Class: Logic
Interview Date: March 18, 1994

1. Overall design problem

The design of a serial protocol converter used in packet videoconferencing (i.e. look at bitstream of a HDLC line and convert it to PictureTel format).



(*Note*: This board was somewhat different than other designs. ICs could be easily removed and an empty backup socket was available in case EPLD density was exceeded. The EPLD was approximately 95% loaded. In addition, no data sheets about the PictureTel protocol was available; the design required extensive trial and error.)

2. Description of error

During a conversion, a CRC (cyclical redundancy check) is performed. The CRC has a property that cannot be found in data books (e.g. the bit ordering sequence was shifted out in a certain way) and AK's assumptions with respect to bit ordering was incorrect.

3. Description of error cause

Assumption was incorrect (correct information was not available in data books).

4. Method of error detection and analysis

AK tried to run a packet through the prototype but it failed to go through. Problem was quickly traced to CRC.

5. Reason for not detecting error during simulation

Simulation was successful but it was based on wrong assumption. AK was unable to find a correct description of CRC properties and therefore couldn't design the appropriate simulation.

6. Method of error correction

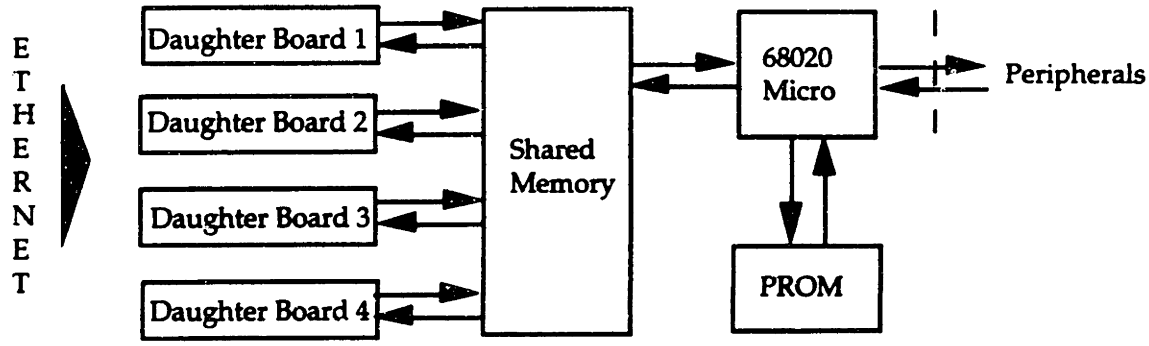
AK varied the assumptions and reprogrammed chip and tested it. If it failed, the trial and error cycle was repeated. It took about 30-40 prototype iterations until the design finally worked.

Error Report Note #3

Error Code: LD-01
Error Class: Logic
Interview Date: March 22, 1994

1. Overall design problem

The design of the main board and four daughter boards for an internet router (project known as BI4).

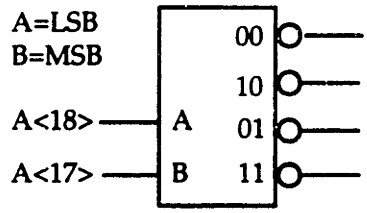


The original designers wanted a high speed internet interface with local processing capability (that project was known as the butterfly interface). The butterfly was designed by a fresh graduate who ended up developing a very complicated board running at 60 MHz (name BI4). The design failed during a demo and LD (+ another experienced designer) were asked to fix it. They were able to get it to work by patching the design with numerous wires which made it volume production nearly impossible. Without formal approval, LD and the other designer spent 3 weeks redesigning the BI4.
(The design didn't use EPLDs. Instead, it used PALs that had the density of 1/20 of a modern Altera. Simulation tools were similar but computers were slower. Design tools were not completely bug-free.)

2. Description of error

The lines for MSB (most significant bit) and LSB (least signficant bit) in a 2-to-4 decoder were accidentally switched during the design. Thus, the case

00 and 11 were not affected (board 1 and 4 were correctly addressed) but the addresses for boards 2 and 3 were switched.



2 to 4 decoder

3. Description of error cause

Oversight during design.

4. Method of error detection and analysis

When the daughter card was inserted in slot #2, it didn't work. The decoder addressed the wrong slot. The error was easily caught with the help of probes and a logic analyzer. (Note: The PAls had not internal states and were easy to probe which made the analysis much easier.)

5. Reason for not detecting error during simulation

LD only simulated the first case (00) and assumed that the other three cases were identical. He took a calculated decision not to simulate the other cases by weighing the incremental effort to do so against the risk adjusted effort to detect, analyze, and fix a possible error. Thus simulation would have caught the error if LD had decided to simulate.

6. Method of error correction

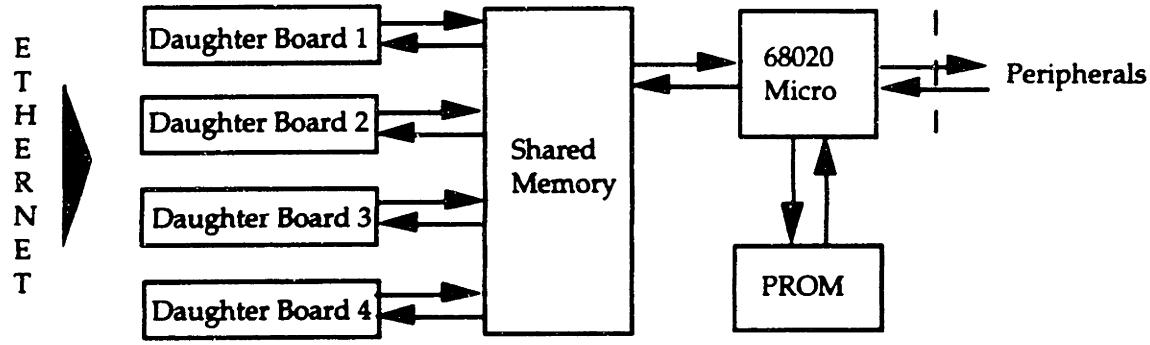
The two pins (A and B) were lifted and two wires were used to reverse them.

Error Report Note #4

Error Code: LD-02
Error Class: Timing
Interview Date: March 22, 1994

1. Overall design problem

The design of the main board and four daughter boards for an internet router (project known as BI4).

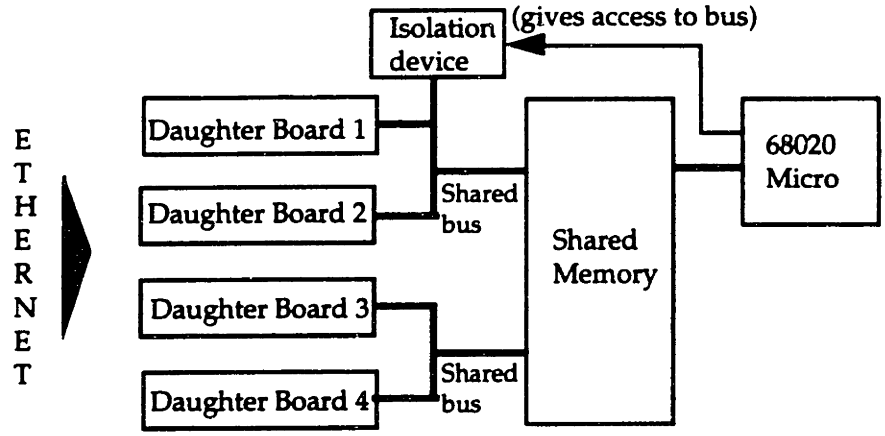


The original designers wanted a high speed internet interface with local processing capability (that project was known as the butterfly interface). The butterfly was designed by a fresh graduate who ended up developing a very complicated board running at 60 MHz (name BI4). The design failed during a demo and LD (+ another experienced designer) were asked to fix it. They were able to get it to work by patching the design with numerous wires which made it volume production nearly impossible. Without formal approval, LD and the other designer spent 3 weeks redesigning the BI4.
(The design didn't use EPLDs. Instead, it used PALs that had the density of 1/20 of a modern Altera. Simulation tools were similar but computers were slower. Design tools were not completely bug-free.)

2. Description of error

The design was planned for 4 ethernet connections with a shared bus. Had to tell ethernet that it owned the entire bus but once ethernet was on it, it couldn't back off. If the 68020 wanted access to the bus, it had to signal

ethernet with the help of an isolation device. Ethernet would have to acknowledge the request and vacate the bus. The error: at the tail end of the request (end of signal), there was a point where both sides disagreed with respect to availability of registers at ethernet controller (there was subtle overlap between timing signals). This error occurred approximately every 20 minutes of actual run time on a loaded system.



3. Description of error cause

Subtle timing interaction between different parts of the board. Original design looked ok.

4. Method of error detection and analysis

When the system was turned on, the outcoming data was junk. The error was infrequent (once in 20 min.) and the designer(s) had no idea where to insert the trigger point (i.e. where to probe and at what clock cycle to probe). Furthermore, the error had a delay which made the analysis even more difficult. They went back to the original design and developed theories about a probable cause. Whenever they had a theory, they went to the lab and tried to confirm or refute their theory by probing with the logic analyzer. It took a long time to find the error.

5. Reason for not detecting error during simulation

It would have been almost impossible to detect the error with simulation because (1) error occurred once in 20 min. of actual run time which would take years to simulate; (2) simulation model was too primitive.

6. Method of error correction

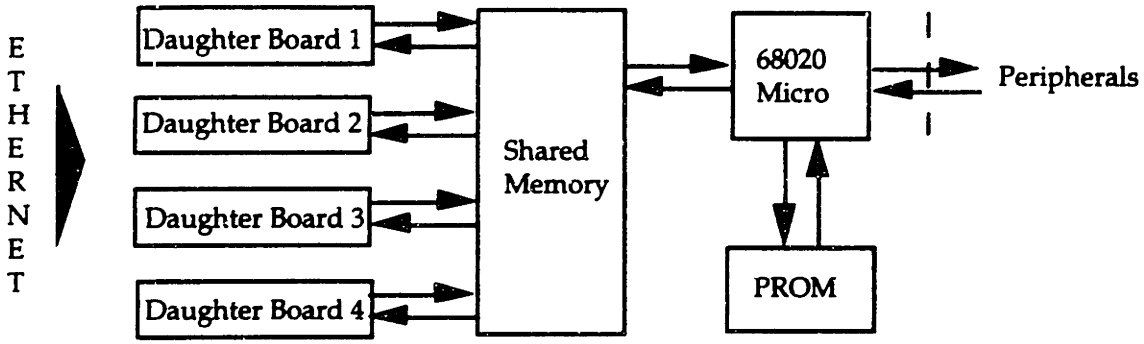
Reprogrammed PAL with additional time delay.

Error Report Note #5

Error Code: LD-03
Error Class: Timing
Interview Date: March 22, 1994

1. Overall design problem

The design of the main board and four daughter boards for an internet router (project known as BI4).



The original designers wanted a high speed internet interface with local processing capability (that project was known as the butterfly interface). The butterfly was designed by a fresh graduate who ended up developing a very complicated board running at 60 MHz (name BI4). The design failed during a demo and LD (+ another experienced designer) were asked to fix it. They were able to get it to work by patching the design with numerous wires which made it volume production nearly impossible. Without formal approval, LD and the other designer spent 3 weeks redesigning the BI4.
(The design didn't use FPGAs. Instead, it used PALs that had the density of 1/20 of a modern Altera. Simulation tools were similar but computers were slower. Design tools were not completely bug-free.)

2. Description of error

The designed used a flash memory located on the peripheral bus (*aka* the slow bus, because it was the slowest bus on the board). The slow bus had a relatively high capacitance which is normally not a problem. The flash

memory was driving the slow bus but it did not have enough output drive capacity.

3. Description of error cause

While the data sheets correctly stated the low output drive capacity, LD did not consult it. Flash memories had been used before successfully in similar designs and the bus capacitance was well within reasonable limits. Normally the memory should be able to drive the bus. The assumption was reasonable but incorrect.

4. Method of error detection and analysis

Someone else ran a prototype. Looked at oscilloscope, measured rise and fall time of signal and it was obvious that it took too long. Looked up flash memory datasheet and cause was found.

5. Reason for not detecting error during simulation

Simulation tools don't consider line capacitances. Couldn't simulate (even if he could, he may have missed it.)

Error Report Note #6

Error Code: LD-04
Error Class: Signal quality
Interview Date: March 30, 1994

1. Overall design problem

The design of a hydrophonic data acquisition card for an underwater location measurement system (project known as LMSHSU).

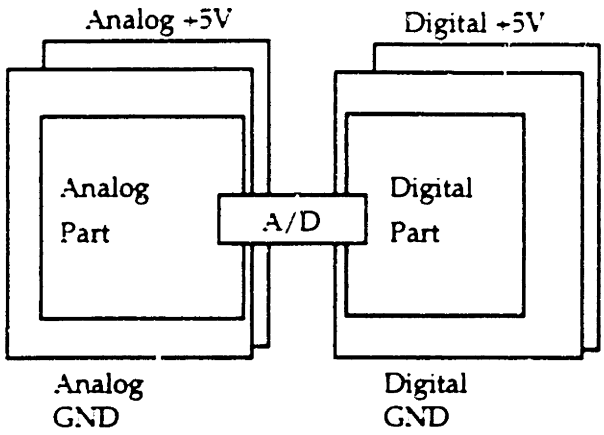
The Navy was developing instrumentation to pick up accoustic information using underwater hydrophonic devices. In order to test it, they immersed a large metal container at the bottom of a lake and blew it up. The duration of data acquisition was 20 sec. as there was some variation with respect to trigger delays. The [analog] data was fed into a data acquisition board which converted it into digital form and stored it in memory. The host computer could then pick up the data and analyze it.

Since standard A/D cards didn't have the right sample rates and frequency filters, LD's group was asked to design a custom card. It ended up having two programmable sampling rates, 4 channels of acquisition, 16 bit resolution, a 20 sec. sampling window, and lots of dynamic memory.

The digital part of the design was partially simulated; the analog part was designed by Prof. Roberge who is a well-known expert in analog design. The analog design was not hard in terms of design structure but in order to achieve noise robustness, the designer had to be an expert in picking the right component ("the right op.amp. out of 50,000 available op.amps"). Furthermore, it was the first time that LD used an Actel EPLD. Part of the board layout was done by a junior designer.

2. Description of error

Since the A/D converter had a resolution of 16 bits, it was extremely sensitive to noise (e.g. a signal of 1V could be distorted by noise in the mV range). Therefore they needed quiet power supplies, i.e. the power supplies had to be separated from the signal. This was achieved by partitioning the GND planes and the +5V planes (see below).



The junior designer partitioned the GND planes correctly, but failed to partition +5V even though he was told to do so. As a result, switching noise was introduced into the signal.

3. Description of error cause

Sloppiness ("didn't do what he was supposed to") and inexperience ("failed to foresee problem") on part of the junior designer.

4. Method of error detection and analysis

Ran the board in the lab. When 0V was applied on the analog side, they got 4 bits at the output which was a clear indication of noise. They looked at the design layout and immediately discovered the problem ("it was something that a good designer would never do").

5. Reason for not detecting error during simulation

Cannot be simulated with existing simulators.

6. Method of error correction

They should have redone the design with partitioned +5V planes. Because of time and cost constraints, however, they were forced to try [inexpensive and fast] ways of noise reduction (capacitors, coax, etc.).

Error Report Note #7

Error Code: LD-05
Error Class: Logic
Interview Date: March 30, 1994

1. Overall design problem

The design of a hydrophonic data acquisition card for an underwater location measurement system (project known as LMSHSU).

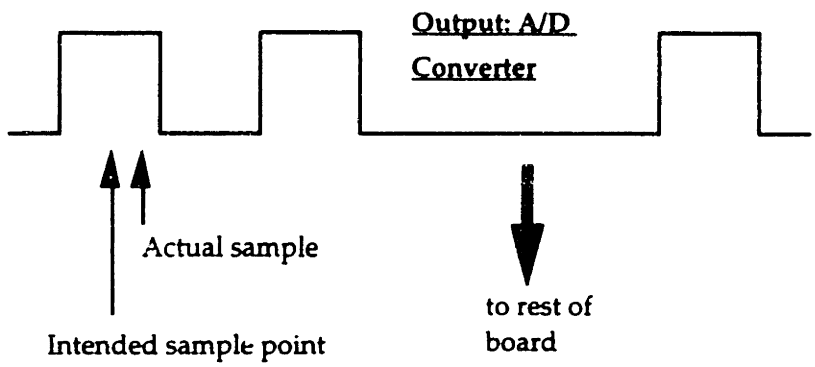
The Navy was developing instrumentation to pick up accoustic information using underwater hydrophonic devices. In order to test it, they immersed a large metal container at the bottom of a lake and blew it up. The duration of data acquisition was 20 sec. as there was some variation with respect to trigger delays. The [analog] data was fed into a data acquisition board which converted it into digital form and stored it in memory. The host computer could then pick up the data and analyze it.

Since standard A/D cards didn't have the right sample rates and frequency filters, LD's group was asked to design a custom card. It ended up having two programmable sampling rates, 4 channels of acquisition, 16 bit resolution, a 20 sec. sampling window, and lots of dynamic memory.

The digital part of the design was partially simulated; the analog part was designed by Prof. Roberge who is a well-known expert in analog design. The analog design was not hard in terms of design structure but in order to achieve noise robustness, the designer had to be an expert in picking the right component ("the right op.amp. out of 50,000 available op.amps"). Furthermore, it was the first time that LD used an Actel EPLD. Part of the board layout was done by a junior designer.

2. Description of error

When the signal comes out of the A/D converter, the acquisition board picks it up at specific points in time. The actual sample point was displaced by one cycle.



3. Description of error cause

Made an error in designing timing (probably oversight).

4. Method of error detection and analysis

Ran prototype and looked at sampling point using oscilloscope; noticed the displaced sampling point. Cause was obvious from the observation of error.

5. Reason for not detecting error during simulation

Combination of many things: poor simulation tools, first time use of Actel, project was somewhat in disarray, etc. Under normal conditions (better tools, more time), simulation should have been able to find error.

6. Method of error correction

Fixed error inside EPLD.

Error Report Note #8

Error Code: LD-06
Error Class: Logic
Interview Date: March 30, 1994

1. Overall design problem

The design of a hydrophonic data acquisition card for an underwater location measurement system (project known as LMSHSU).

The Navy was developing instrumentation to pick up accoustic information using underwater hydrophonic devices. In order to test it, they immersed a large metal container at the bottom of a lake and blew it up. The duration of data acquisition was 20 sec. as there was some variation with respect to trigger delays. The [analog] data was fed into a data acquisition board which converted it into digital form and stored it in memory. The host computer could then pick up the data and analyze it.

Since standard A/D cards didn't have the right sample rates and frequency filters, LD's group was asked to design a custom card. It ended up having two programmable sampling rates, 4 channels of acquisition, 16 bit resolution, a 20 sec. sampling window, and lots of dynamic memory.

The digital part of the design was partially simulated; the analog part was designed by Prof. Roberge who is a well-known expert in analog design. The analog design was not hard in terms of design structure but in order to achieve noise robustness, the designer had to be an expert in picking the right component ("the right op.amp. out of 50,000 available op.amps"). Furthermore, it was the first time that LD used an Actel EPLD. Part of the board layout was done by a junior designer.

2. Description of error

The Actel EPLD has a feature that allows designers to probe internal states with the help of additional equipment (that Actel will readily sell). The feature is activated by a mode pin and then uses 4 of the available pins for access to internal states. LD decided not to use the probing feature and used the four pins as part of his design. Since he didn't use the feature, he left the

mode pin unassigned (float), thinking that the default is "no probe". Instead the default was "probe" and their EPLD was set in the probing mode.

3. Description of error cause

Wrong assumption.

4. Method of error detection and analysis

Turned on prototype and noticed that only a few signals worked. The cause was found by iterating between signal probing and looking at data sheets. When LD arrived at the mode pin description, the cause was clear. Took about one day to find cause.

5. Reason for not detecting error during simulation

There were two reasons: (1) LD was writing his own simulation model and since his assumption was incorrect, it would not have been able to correctly specify it in the simulation model; and (2) including the mode pin in simulation is extremely complicated and also Actel specific.

6. Method of error correction

Tied mode pin to ground (turned probing mode off).

Error Report Note #9

Error Code: LD-07
Error Class: Logic
Interview Date: March 30, 1994

1. Overall design problem

The design of a hydrophonic data acquisition card for an underwater location measurement system (project known as LMSHSU).

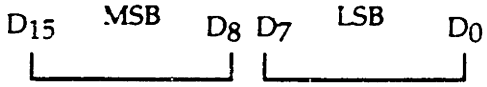
The Navy was developing instrumentation to pick up accoustic information using underwater hydrophonic devices. In order to test it, they immersed a large metal container at the bottom of a lake and blew it up. The duration of data acquisition was 20 sec. as there was some variation with respect to trigger delays. The [analog] data was fed into a data acquisition board which converted it into digital form and stored it in memory. The host computer could then pick up the data and analyze it.

Since standard A/D cards didn't have the right sample rates and frequency filters, LD's group was asked to design a custom card. It ended up having two programmable sampling rates, 4 channels of acquisition, 16 bit resolution, a 20 sec. sampling window, and lots of dynamic memory.

The digital part of the design was partially simulated; the analog part was designed by Prof. Roberge who is a well-known expert in analog design. The analog design was not hard in terms of design structure but in order to achieve noise robustness, the designer had to be an expert in picking the right component ("the right op.amp. out of 50,000 available op.amps"). Furthermore, it was the first time that LD used an Actel EPLD. Part of the board layout was done by a junior designer.

2. Description of error

The acquisition board had a feature to generate an interrupt on the CPU it was connected to. On a VME bus all interrupts have to be vectored (i.e. an interrupt is first sent to the CPU, the CPU acknowledges the interrupt and requests a vector with the address). The interrupt vector is a two byte word of the following format:



The MSB is used for the CPU vector whereas the LSB is unused (i.e. it can contain anything). LD thought that the address is on the LSB.

3. Description of error cause

Wrong assumption.

4. Method of error detection and analysis

Turned on prototype and initiated an interrupt. The CPU ended up in a bogus location and got stuck. The problem was found by reviewing the design and looking at the specification sheets. It took a few days to find the problem.

5. Reason for not detecting error during simulation

LD wrote the simulation model for the VME bus (i.e. the wrong assumption was built into the simulation).

6. Method of error correction

The LSB was copied into the MSB.

Error Report Note #10

Error Code: LD-08
Error Class: Logic
Interview Date: March 30, 1994

1. Overall design problem

The design of a switch card for an asynchronous transfer mode (ATM) networking box/switch (named the Emerald project). The Emerald connects wide area networks (WAN) to the ethernet, X.25 packet switch networks, token rings, and other WANs. It is a multi-board system where the data flows from the interface to other interface cards (up to 12) over an active element - the switch card. The switch has to connect the right interfaces at the right point in time.

2. Description of error

On the switch card was a test and control subsystem which would do things like self-tests and redundancy tests. It had a microprocessor that was connected to a few peripheral devices. In order to connect the devices, a decoder was designed into an EPLD. When the decoder equations were written, one of the select lines ended up active high instead of active low. As a result, one of the select lines was inverted.

3. Description of error cause

Oversight in the way the logic equations were written.

4. Method of error detection and analysis

LD doesn't know; another engineer did it. LD remembered that the engineer found it very quickly (easy analysis).

5. Reason for not detecting error during simulation

Didn't simulate all cases (i.e. there were similar test cases that worked well in simulation). LD figured that the probability of an error was very small. But mainly, the EPLD was not full (sufficient space for corrections left)

and all inputs/outputs were routed through the EPLD. Thus LD felt that an error would be easy to find and fix.

6. Method of error correction

Don't know (probably reprogram EPLD).

Error Report Note #11

Error Code: PC-01
Error Class: Timing
Interview Date: March 22, 1994

1. Overall design problem

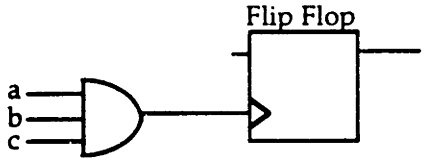
The design of a ground terminal for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard).

PC reported on the redesign of the network processor which was completed by a graduate student. The old design was done using lower density EPLDs and the student was supposed to upgrade the design into higher density EPLDs. According to PC, a less experienced designer was selected because the design was "straight-forward".

The project is internally known as ACTS.

2. Description of error

Part of the design involved loading a latch (which sets off a chain of events). The signal to load the latch was given by three statebits (i.e. they had to be 111). The state bits were internal to an EPLD and the interpretation was done in a macrocell.



Because the flip flop was not clocked, it would be set off whenever the three state bits where 111 (asynchronously). During state transitions, however, bits sometimes go through intermediate transient states that are difficult to predict because they arise due to the complex behavior of the circuit. For example, a transition from 110 to 101 does not involve 111 and should not trigger a latch load. In reality, however, the transition may work like this: 110 to 111 to 101. The initial and final states are identical but the intermediate

state triggers a latch load. This is exactly what happened in this design (the error occurred every two hours).

3. Description of error cause

Designer did not foresee possible problem and tried to implement "elegant" solution instead of applying prudent design rules.

4. Method of error detection and analysis

The software people ran the operating system and it died on them (they hit the <CR> key and nothing happened). After verifying the software, it appeared to be a hardware problem. The software people generated a pulse which was traced by hardware people in order to find a trigger point. Once they found a trigger point and localized the problem to an EPLD, the problem was found by just looking at the design.

5. Reason for not detecting error during simulation

The error couldn't be detected for two reasons: (1) it occurred every two hours of real time execution; (2) timing simulators have only limited capability in simulating complex timing patterns (e.g. only maximum and minimum delays can be specified; intermediate transients are impossible to simulate).

6. Method of error correction

The flip flop was clocked (synchronous design) which eliminated transients between clock cycles.

Error Report Note #12

Error Code: PC-02
Error Class: Logic
Interview Date: April 1, 1994

1. Overall design problem

The design of a terminal controller for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). The project is internally known as ACTS.

PC was working on the burst modem interface which is the most difficult design in the ACTS project. The interface is a combo of 12 EPLDs and 50 ECL (emitter-current logic) chips. It uses multiple frequencies (high: 174 MHz; low: 29 MHz) and has additional 300K-transistor gate arrays on it (chips used for error correction and detection). In sum, the board contains approximately 90 chips and has both, a digital (simulated) and an analog (not simulated) section. It took PC about three months (at 3/4 capacity) to design and simulate the board. To him, the board was not extremely difficult but it was no cakewalk either.

The board had hundreds of bugs that were mostly detected in simulation. About 20% of the bugs had to do with filling up the EPLD (design wouldn't fit) and routing it (wouldn't route). The EPLDs were on the average 80% full. About 30% of the bugs had to do with the two speeds of operation (switch between speeds). The remaining errors had to do with standard human oversights and the subtle interaction between multiple finite state machines. (They are developed concurrently and exchange signals; typical problems: timing, communication, deadlock).

The design is 95% completed (some specification changes just came in) and had no logic errors in the prototype.

2. Description of error

The design had two op. amps and one of the op.amps had the inputs reversed (inverting and non-inverting inputs).

3. Description of error cause

Oversight.

4. Method of error detection and analysis

When the prototype was tested, the outputs were not right; one of the outputs rails was frozen (pegged at maximum). The error was found quickly by (1) looking at schematic for possible causes; and (2) comparing it with prototype.

5. Reason for not detecting error during simulation

Did not simulate analog section.

6. Method of error correction

Lifted pins and connected crossed wires.

Error Report Note #13

Error Code: PC-03
Error Class: Signal Quality
Interview Date: April 1, 1994

1. Overall design problem

The design of a terminal controller for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). The project is internally known as ACTS.

PC was working on the burst modem interface which is the most difficult design in the ACTS project. The interface is a combo of 12 EPLDs and 50 ECL (emitter-current logic) chips. It uses multiple frequencies (high: 174 MHz; low: 29 MHz) and has additional 300K-transistor gate arrays on it (chips used for error correction and detection). In sum, the board contains approximately 90 chips and has both, a digital (simulated) and an analog (not simulated) section. It took PC about three months (at 3/4 capacity) to design and simulate the board. To him, the board was not extremely difficult but it was no cakewalk either.

The board had hundreds of bugs that were mostly detected in simulation. About 20% of the bugs had to do with filling up the EPLD (design wouldn't fit) and routing it (wouldn't route). The EPLDs were on the average 80% full. About 30% of the bugs had to do with the two speeds of operation (switch between speeds). The remaining errors had to do with standard human oversights and the subtle interaction between multiple finite state machines. (They are developed concurrently and exchange signals; typical problems: timing, communication, deadlock).

The design is 95% completed (some specification changes just came in) and had no logic errors in the prototype.

2. Description of error

Another designer was in charge of the power supply (needed special design since multiple voltages were required). She designed the supply

correctly but somehow the board had two wires that touched each other (were supposed to be separated). It is unclear how that happened.

3. Description of error cause

Unknown (perhaps during layout transfer).

4. Method of error detection and analysis

When the prototype was tested, some of the voltages were incorrect. With the help of a voltmeter, the error was localized and by physically inspecting the prototype, the cause was found quickly.

5. Reason for not detecting error during simulation

Couldn't simulate (error probably happened between simulation and prototyping).

6. Method of error correction

PC doesn't remember (lines were probably separated on the board).

Error Report Note #14

Error Code: PC-04
Error Class: Timing
Interview Date: April 1, 1994

1. Overall design problem

The design of a terminal controller for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). The project is internally known as ACTS.

PC was working on the burst modem interface which is the most difficult design in the ACTS project. The interface is a combo of 12 EPLDs and 50 ECL (emitter-current logic) chips. It uses multiple frequencies (high: 174 MHz; low: 29 MHz) and has additional 300K-transistor gate arrays on it (chips used for error correction and detection). In sum, the board contains approximately 90 chips and has both, a digital (simulated) and an analog (not simulated) section. It took PC about three months (at 3/4 capacity) to design and simulate the board. To him, the board was not extremely difficult but it was no cakewalk either.

The board had hundreds of bugs that were mostly detected in simulation. About 20% of the bugs had to do with filling up the EPLD (design wouldn't fit) and routing it (wouldn't route). The EPLDs were on the average 80% full. About 30% of the bugs had to do with the two speeds of operation (switch between speeds). The remaining errors had to do with standard human oversights and the subtle interaction between multiple finite state machines. (They are developed concurrently and exchange signals; typical problems: timing, communication, deadlock).

The design is 95% completed (some specification changes just came in) and had no logic errors in the prototype.

2. Description of error

The design has a T_X offset 12 bit register that tracks the distance from the satellite and is used as an offset during operation (by x number of clock cycles). PC tried to put the register inside the EPLD but couldn't fit them .

Instead, he used a part of the EPLD called expander latch (a NAND with 36 inputs) which is normally not used because it results in a asynchronous solution. The signal was loaded from a macrocell and probably its width was too narrow and the pulse was swallowed (total pulse width was too narrow relative to the clock cycle). Reason: probably asynchronous nature of register.

3. Description of error cause

PC wasn't aware of possible consequences of his design solution.

4. Method of error detection and analysis

When the prototype was tested, they couldn't load the T_X register. PC tried to recreate error in simulator but was unsuccessful. Probing and design inspection led him to make a design change and the error went away. Took 2-3 hours to find.

5. Reason for not detecting error during simulation

Simulator cannot capture subtle timing variation. For example, a 1 ps pulse on a 20ns delay will be probably be swallowed in a design but simulator will treat the pulse as undegradated.

6. Method of error correction

Used synchronous cell to widen pulse to 35 ns.

Error Report Note #15

Error Code: PC-05
Error Class: Signal Quality
Interview Date: April 1, 1994

1. Overall design problem

The design of a terminal controller for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). The project is internally known as ACTS.

PC was working on the burst modem interface which is the most difficult design in the ACTS project. The interface is a combo of 12 EPLDs and 50 ECL (emitter-current logic) chips. It uses multiple frequencies (high: 174 MHz; low: 29 MHz) and has additional 300K-transistor gate arrays on it (chips used for error correction and detection). In sum, the board contains approximately 90 chips and has both, a digital (simulated) and an analog (not simulated) section. It took PC about three months (at 3/4 capacity) to design and simulate the board. To him, the board was not extremely difficult but it was no cakewalk either.

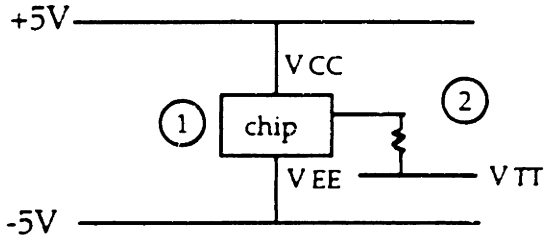
The board had hundreds of bugs that were mostly detected in simulation. About 20% of the bugs had to do with filling up the EPLD (design wouldn't fit) and routing it (wouldn't route). The EPLDs were on the average 80% full. About 30% of the bugs had to do with the two speeds of operation (switch between speeds). The remaining errors had to do with standard human oversights and the subtle interaction between multiple finite state machines. (They are developed concurrently and exchange signals; typical problems: timing, communication, deadlock).

The design is 95% completed (some specification changes just came in) and had no logic errors in the prototype.

2. Description of error

Another designer was asked to come up with a board design for PECL logic. In standard logic, designers always insert capacitors that drive current spikes in the logic part (power supplies are normally too slow to drive these

spikes). Since this case did not involve standard logic, she decided to leave out capacitors altogether (PC doesn't understand why). After the error was accidentally discovered, she was asked to put in capacitors but she ended up putting them in the wrong place.



There are supposed to be two types of capacitors that take care of current spikes (1 and 2 in figure above).

3. Description of error cause

The designer didn't understand PECL and failed to see the consequences of her design deficiencies.

4. Method of error detection and analysis

When the prototype was tested, the error was discovered accidentally during one of the schematic reviews. In general, it would have been difficult to detect error since noise problems are difficult to detect (they are stochastic and only occur under certain noise conditions.)

5. Reason for not detecting error during simulation

Cannot simulate current spikes in PECL.

6. Method of error correction

It took 2 weeks to correct the problem and almost resulted in a complete respin of the board.

Error Report Note #16

Error Code: PC-06
Error Class: Logic
Interview Date: April 8, 1994

1. Overall design problem

The design of a terminal controller for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). The project is internally known as ACTS.

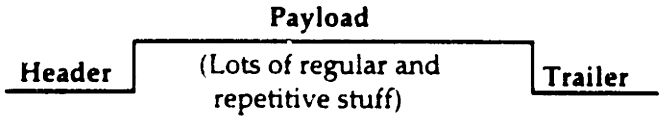
PC was working on the burst modem interface which is the most difficult design in the ACTS project. The interface is a combo of 12 EPLDs and 50 ECL (emitter-current logic) chips. It uses multiple frequencies (high: 174 MHz; low: 29 MHz) and has additional 300K-transistor gate arrays on it (chips used for error correction and detection). In sum, the board contains approximately 90 chips and has both, a digital (simulated) and an analog (not simulated) section. It took PC about three months (at 3/4 capacity) to design and simulate the board. To him, the board was not extremely difficult but it was no cakewalk either.

The board had hundreds of bugs that were mostly detected in simulation. About 20% of the bugs had to do with filling up the EPLD (design wouldn't fit) and routing it (wouldn't route). The EPLDs were on the average 80% full. About 30% of the bugs had to do with the two speeds of operation (switch between speeds). The remaining errors had to do with standard human oversights and the subtle interaction between multiple finite state machines. (They are developed concurrently and exchange signals; typical problems: timing, communication, deadlock).

The design is 95% completed (some specification changes just came in) and had no logic errors in the prototype.

2. Description of error

When running a simulation, one typically simulates a complete data packet (real time: appoximately. 4ms).



A complete packet consists of a header, payload, and a trailer. While the header (e.g. locations) and the trailer (e.g. checksums) contain random stuff, the payload contains a lot of repetitive operations. The payload is roughly 20,000 clock ticks long and if simulated completely, it would result in a 10 GByte file (with a lot of useless info) and a very long simulation time. Thus, PC resorts to a technique (which seems to be standard) that combines several ticks in a formula and simulates the formula. Instead of simulating 20,000 ticks, he can get away with only 64 (i.e. he economizes on run time).

In this case, PC accidentally reversed a sign in the formula (typing error) and simulated the formula. Coincidentally, the incorrect formula happened to give the correct answer and the simulation passed. In the design, however, one of the counters was 20 ticks short (320 instead of 340) and as a result the data read was started and terminated at the wrong clock cycles (i.e. it was offset).

3. Description of error cause

Counter was off by 20 clock ticks.

4. Method of error detection and analysis

Ran prototype at different speeds and noticed a problem with the data being read. Hooked up a scope and looked at output for different speeds. Went back to simulation printout and compared lab results with simulation. Noticed discrepancy. Checked formula and found error. Took a while to find error because PC was looking for noise problems first.

5. Reason for not detecting error during simulation

Economized on the number of cases being tested. Error slipped through because of typo in formula. Could have detected error if exhaustive simulation had been performed.

6. Method of error correction

Made change in EPLD to reset offset instead of redesigning counter. Added comment that redesign isn't necessary because it wouldn't affect the rest of the design.

Error Report Note #17

Error Code: RP-01
Error Class: Logic
Interview Date: March 18, 1994

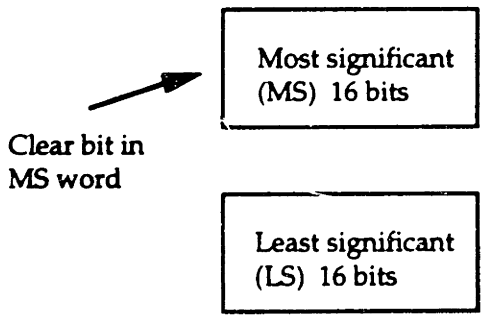
1. Overall design problem

The design of a ground terminal for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard).

RP worked on a queued interrupt controller. The project is internally known as ACTS.

2. Description of error

As part of the design, a multiplexed bus tries to read out some bits from a register. After the read-out is complete, the bit is normally cleared. In this case, however, the bus had access to the wrong word in the register and was unable to clear the correct bit.



Instead of trying to clear the bit in the MS word, the bit tried to clear it in the LS word.

3. Description of error cause

Typing error during IC design.

4. Method of error detection and analysis

Programmers ran their software with the prototype but couldn't clear the bit in question. After a few months, the OS people came up to RP and informed him about the problem. The cause was found by inspecting the relevant portion of the design code.

5. Reason for not detecting error during simulation

There were a total of 16 bits in the register, each undergoing an identical operation. Because designing all test cases would have been tedious and time-consuming, RP took a calculated risk and simulated bus access for only 1-2 bits. Because the first simulations passed without problems, he assumed the rest to be error-free. In case there was an error, he felt that fixing a possible error should be fairly easy. In other words, he expected the incremental effort to correct a possible error to be less than the incremental effort to test all cases times the expected probability of making a mistake.

6. Method of error correction

It turned out that fixing the error took more effort than previously thought. Needed a software workaround to solve problem.

Error Report Note #18

Error Code: RP-02
Error Class: Logic
Interview Date: March 18, 1994

1. Overall design problem

The design of a ground terminal for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard).

RP worked on a queued interrupt controller. The project is internally known as ACTS.

2. Description of error

The bus arbiter determines which master can access the bus and in which order the access occurs. The bus arbiter rules are as follows: WRITE has priority over READ; an access has to be completed before the next access is arbitrated. During heavy traffic conditions, the following error occurred: two masters, one READ and one WRITE, tried to gain access to the bus simultaneously; the WRITE received priority and completed the access; in the meantime another master arrived with a WRITE request; the arbiter again gave priority access to the new WRITE request instead of letting the queued master complete its READ request. If a continuous inflow of WRITE requests took place (heavy traffic), the READ requests was queued indefinitely.

3. Description of error cause

RP overlooked the importance of the described arbiter features.

4. Method of error detection and analysis

Noticed error when physically testing prototype board and playing around with various tests. Analysis was done by probing the chip with the help of an oscilloscope and the error was found easily. (Only 84 pin package with 60% density.)

5. Reason for not detecting error during simulation

The basic simulation was performed but not under the appropriate heavy traffic conditions (assumed incorrect traffic conditions). Simulation should have almost used random test vectors. Under the right test conditions, the error could have been caught but it would have taken a skilled simulator two days to set up the appropriate simulation (this was Ron's first simulation; would have taken longer).

6. Method of error correction

Removed prototype from PC board, changed design, reprogrammed chip and inserted it in prototype. (Note: Removing a prototype can be a very cumbersome process using SMT technology.)

Error Report Note #19

Error Code: RP-03
Error Class: Timing
Interview Date: March 18, 1994

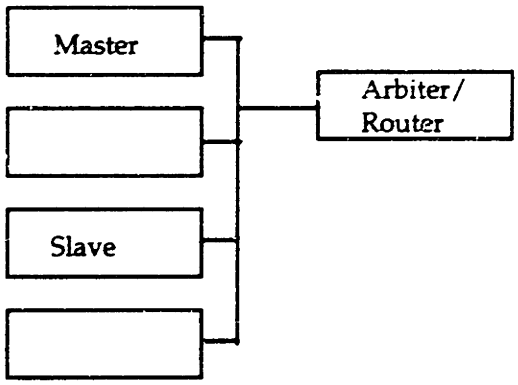
1. Overall design problem

The design of a ground terminal for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard).

RP worked on a bus interface chip. The project is internally known as ACTS.

2. Description of error

When a master pushes an address on the bus, the arbiter/router receives the address and establishes the route from master to slave (address).



Because RP didn't allow the arbiter/router enough time to decode the address, sometimes the incorrect slave was selected.

3. Description of error cause

The assumption about decoding time was right on the edge of performance; no safety margin was included.

4. Method of error detection and analysis

Caught the error while randomly checking loops on the prototype. Once error was detected, RP used two scope probes to find error cause.

5. Reason for not detecting error during simulation

The simulation predicted that timing assumption was sufficient. Simulation, however, is incomplete with respect to predicting timing behavior since some design features are not captured in the models (e.g. length of bus (impedance), pin capacitance, etc.). In essence, reality was more demanding than simulation and therefore RP felt that he should have included a safety margin.

6. Method of error correction

RP gave the arbiter/router one extra clock cycle (which slightly decreased overall speed; no problem, since design was already faster than required). EPLD was reprogrammed.

Error Report Note #20

Error Code: RP-04
Error Class: Logic
Interview Date: March 18, 1994

1. Overall design problem

The design of a ground terminal for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). The project is internally known as ACTS.

2. Description of error

When designing the bus arbiter, RP made an assumption about the rate at which data is fed into the bus (burst transfer rate). One module didn't meet this assumption. (The connection to the bus looks like a FIFO: shift in, shift out, input ready, and output ready. RP assumed that output ready is not checked => assumed rate was higher than actual rate.)

3. Description of error cause

RP thinks that he simulated the design during a first pass and then made change to the module. He probably forgot to resimulate the entire design since it took too much time. Wrong assumption about transfer rate.

4. Method of error detection and analysis

Ron ran tests on prototype; send data into the bus but got shifted data out. Went back to simulation to check his design and found burst rate error.

5. Reason for not detecting error during simulation

Didn't resimulate after change was made.

6. Method of error correction

Changed burst rate at module in question; reprogrammmed EPLD.

Error Report Note #21

Error Code: RP-05
Error Class: Logic
Interview Date: April, 1994

1. Overall design problem

The design of a RAM controller for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). The controller contains microcode controlling the data transfer between optical lines in the satellite. The project is internally known as ACTS.

2. Description of error

The controller contains a feature that takes data from SONET and reblocks it for the satellite. Since there were padding locations available (locations filled with dummy data), RP decided to utilize these locations for additional use such as data exchange for computer terminals (he included two features: (1) a data source identifier; and (2) data piggyback for data exchange). Sometime during the project a specification change was made. Before the change, zero values had no meaning but after the change they were invalid. When RP checked his design for zero values, he overlooked the added feature that took advantage of padding (maybe because it wasn't a design requirement).

3. Description of error cause

Forgot to resimulate feature that wasn't required.

4. Method of error detection and analysis

RP ran tests on prototype; he tried to pass data but it didn't work. He looked at design and found problem quickly.

5. Reason for not detecting error during simulation

Didn't resimulate after change was made.

6. Method of error correction

Easy fix.

Error Report Note #22

Error Code: RP-06
Error Class: Logic
Interview Date: April 8, 1994

1. Overall design problem

The design of a ground terminal for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). RP worked on a bus interface chip. The project is internally known as ACTS.

2. Description of error

The interface has three frame pulses. The first pulse is systemwide and marks the beginning of a new 4ms data cycle and is used as a reference. The other two pulses have offsets with respect to the system-wide pulse. One pulse indicates incoming SONET data, the other pulse indicates outgoing SONET data. RP simulated the design for different counter values (1, 2, 3, ...) and concluded that if it worked for these values, it would ok for the rest. It would have been impossible to simulate all values (the counter was 17 bits long!!!). Unfortunately, the counter value is 0 after a reset but RP thought it would could only go down to 1 and never checked 0. But 0 resulted in a no-frame pulse and the counter started to reverse (0 -> FF ->...).

3. Description of error cause

Forgot that counter could reach state 0 if reset. Economized on simulation.

4. Method of error detection and analysis

RP ran tests on prototype; the design didn't work if it was reset. Since it didn't work in the lab, he went back to simulation. But simulation showed no error. Instead of recreating lab conditions, he relied on his simulation test cases which were insufficient. Therefore it took him a day to find the error

(he thinks that he could have been faster if he hadn't relied on his old simulation cases).

5. Reason for not detecting error during simulation

Didn't simulate all cases (he economized on simulation time).

6. Method of error correction

Changed EPLD to count down to 0 instead of 1.

Error Report Note #23

Error Code: RP-07
Error Class: Logic
Interview Date: April 22, 1994

1. Overall design problem

The design of a ground terminal for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). Ron worked on the SONET line card. The project is internally known as ACTS.

2. Description of error

When the SONET data stream comes in, a pulse is used to identify the proper location in the data stream. The pulse came in the incorrect point, i.e. it was a few clocks off.

3. Description of error cause

Forgot some pipeline stages that have an impact on time characteristics.

4. Methods of error detection and analysis

The prototype was run in the lab and pulse came in at wrong time. Technician found the error quickly (one hour) by using probe.

5. Reason for not detecting error during simulation

Didn't simulate the interface to PMC (could have written behavioral model but would have taken too long).

6. Method of error correction

Reprogrammed EPLD (added 2 flops to delay pulse). Took one hour. Even though EPLD was 95% full, it turned out to be an easy fix.

Error Report Note #24

Error Code: RP-08
Error Class: Logic
Interview Date: April 25, 1994

1. Overall design problem

The design of a ground terminal for a communications satellite which functioned as the interface between the satellite and SONET (fiber optic communications standard). Ron worked on the SONET line card. The project is internally known as ACTS.

2. Description of error

Data is passed from SONET through the board to a switch. Data that came out was different than RP expected (address sequencer problem). He first thought that microcode was wrong since the address sequencer is controlled by microcode.

3. Methods of error detection and analysis

Total run time for one data block was 4 ms. RP simulated the first 0.25ms which took the simulator 40 minutes in actual run time. In addition, the simulator wasn't available for testing 4ms until two days later. The error was noticed during a prototype test. A lot of time was spent with the logic analyzer and another simulation revealed that the error occurred after 0.75ms of real time operation.

4. Reason for not detecting error during simulation

Simulator would have detected the error if the entire 4ms had been simulated. RP economized on simulation time since partial testing found no errors.

5. Method of error correction

Reprogrammed EPLD since code inside EPLD was well-structured.

Appendix D

Second Field Study: The EPLD and ASIC Questionnaires

On the following pages, you will find the two questionnaires that were sent to 500 EPLD designers and 500 ASIC designers on August 12, 1994.

D.1 The EPLD Questionnaire

(Please turn to next page.)

MASSACHUSETTS INSTITUTE OF TECHNOLOGY (M.I.T.)

DEPARTMENTS OF ELECTRICAL ENGINEERING AND MANAGEMENT

The M.I.T. Circuit Design Survey

Questionnaire Instructions

In completing this questionnaire, we are asking your cooperation to participate in an important national study designed to obtain a better understanding of design in general, and the design of circuit boards containing electrically-programmable logic devices (EPLDs) in particular. You were carefully selected from a large pool of very qualified design engineers and it is of utmost importance to our research that you respond (a stamped return envelope is enclosed).

If you do not design circuit boards containing EPLDs (e.g. FPGAs, PLDs, etc.), please pass the survey on to someone who does. If you cannot complete the questionnaire and you cannot pass it on to someone else, please write "*wrong experience*" on the questionnaire and send it back to us.

Interested in winning a MIT T-shirt? We will hold a lottery at which *three* designers will receive a MIT T-shirt (see section below).

Thank you for your cooperation !

Stefan Thomke, Research Assistant
Professor Eric von Hippel, Research Director

ENTRY SECTION FOR T-SHIRT LOTTERY

[ ] Yes, I would like to participate in the MIT T-shirt lottery. My size is: S M L XL
(circle one)

My name is: _____
(last name) (first name)

PART 1: YOUR BACKGROUND AND EXPERIENCE

This part will help us to learn about your experience.

(1) How long have you been a circuit designer? _____ years and _____ months.

(2) How long have you been a user of design simulation tools? _____ years and _____ months.

	No, not at all						Yes, absolutely
(3) When you design circuits containing EPLDs, would you describe your design style as incremental (i.e. design a little, test a little, and so on)?	1	2	3	4	5	6	7

PART 2: THE EFFECTIVENESS OF SIMULATION AND PROTOTYPING IN DEBUGGING ERRORS

In the following questions, you will be asked about the general use and effectiveness of **computer simulation** and **prototyping** in debugging errors in circuits containing EPLDs. The following definitions will be used:

- ***Logic errors:*** errors due to faulty logic behavior of design (*examples*: reversed select lines of a decoder; negation of an input signal).
- ***Timing errors:*** errors due to faulty temporal behavior of design (*examples*: insufficient setup and hold times).
- ***Signal quality errors:*** errors due to degradation of signal quality during signal transmission (*example*: introduction of noise in high frequency applications due to excessive line proximity on printed circuit board).

Which is more effective: simulation or prototyping?	***simulation*** more effective			no difference		***prototype*** more effective		do not know
Questions about logic errors								
Which method is more effective in...								
• detecting all logic errors if you could run all possible test cases (i.e. which method is more accurate) ?	1	2	3	4	5	6	7	[ ]
• the analysis of logic errors and finding the error cause?	1	2	3	4	5	6	7	[ ]
• fixing logic errors (and verifying the fix)?	1	2	3	4	5	6	7	[ ]
Questions about timing errors								
Which method is more effective in...								
• detecting all timing errors if you could run all possible test cases (i.e. which method is more accurate)?	1	2	3	4	5	6	7	[ ]
• the analysis of timing errors and finding the error cause?	1	2	3	4	5	6	7	[ ]
• fixing timing errors (and verifying the fix)?	1	2	3	4	5	6	7	[ ]
Questions about signal quality errors								
Which method is more effective in...								
• detecting all signal quality errors if you could run all possible test cases (i.e. which method is more accurate)?	1	2	3	4	5	6	7	[ ]
• the analysis of signal quality errors and finding the error cause?	1	2	3	4	5	6	7	[ ]
• fixing signal quality errors (and verifying the fix)?	1	2	3	4	5	6	7	[ ]

PART 3: SWITCHING FROM COMPUTER SIMULATION TO THE FIRST PROTOTYPE

This part should help us to learn some specifics about the last hardware design that you were involved in.

- Please think back to the ***last*** design:

  (a) that involved the usage of electrically-programmable logic devices (EPLDs) (e.g. FPGA, PLDs)

  AND

  (b) during which you were involved in deciding when to switch from computer simulation to the first hardware prototype.

  This design will be referred to as "your last design" from this point forward.

(1) Please indicate when your *last* design was completed: completed _____ months ago.

(2) Please indicate the number of EPLDs used in your *last* design: _____ EPLDs (incl. repeated EPLDs)

(3) Please indicate the type of EPLDs used in your *last* design: [ ] SRAMs [ ] (E)EPROM
[ ] Antifuses [ ] Others:_____

(4) Please indicate the EPLD manufacturer: [ ] Xilinx [ ] Altera [ ] Others: _____
[ ] Actel [ ] TI _____

(5) Please estimate the total size of your *last* design: approx. _____ gates (incl. copied cells)
(*If* you count in *macrocells*, please multiply the number of macrocells by the gate count per macrocell)

(6) Please estimate the percentage of your design that was copied (e.g. repeated cells): _____ % of total gates

(7) Please indicate the total number of boards that were or will be fabricated: approx. _____ boards

	completely analog						completely digital
(8) Please indicate if your *last* design was.....	1	2	3	4	5	6	7

(9) The following list contains general statements about your *last* design.

Please indicate the accuracy of each statement.	not at all accurate			somewhat accurate		very accurate		cannot answer
• I have done very similar designs before.....	1	2	3	4	5	6	7	[ ]
• The design's run speed was pushed to its absolute limit.....	1	2	3	4	5	6	7	[ ]
• The design had a high degree of logic complexity.....	1	2	3	4	5	6	7	[ ]
• Before starting to simulate, I knew what type of errors (i.e. logic, timing, and signal) the design would have and what their relative distribution would be.....	1	2	3	4	5	6	7	[ ]
• Before starting to simulate, I knew how many errors of each error type the design would have.....	1	2	3	4	5	6	7	[ ]

If you knew something about the type, the distribution and number of errors before starting simulation, please explain why: _____

_____

(10) Did you simulate your design **on paper** before switching to computer simulation or prototyping?..... [ ] Yes [ ] No (if no, skip to (11))

Please indicate the amount of paper simulations done...	very few						very many
• before schematic capture.....	1	2	3	4	5	6	7
• after schematic capture.....	1	2	3	4	5	6	7

(11) Think of **the first time** you decided to switch from **computer simulation to prototyping** (i.e. *the first prototype*). The following questions relate to your **reasons for stopping simulation** and switching to prototyping.

(11a) *About logic errors:*

• I used computer simulation to find logic errors before switching to the first hardware prototype.

[ ] **Yes**

[ ] **No**. Please explain why not: _____

• (if **no**, skip to **11b**) •

• I stopped simulating because I was extremely confident that there were no remaining logic errors in the design..... [ ] Yes (if **yes**, skip to **11b**) [ ] No

I thought that there could be remaining logic errors..	not at all accurate			somewhat accurate			very accurate	do not know
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.....	1	2	3	4	5	6	7	[ ]
• but the rate of logic error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining logic errors.....	1	2	3	4	5	6	7	[ ]
• but I felt that it would be easy to fix the remaining logic errors in the hardware prototype.....	1	2	3	4	5	6	7	[ ]
• Other: _____	1	2	3	4	5	6	7	[ ]

(11b) *About timing errors:*

• I used computer simulation to find timing errors before switching to the first hardware prototype.

[ ] **Yes**

[ ] **No**. Please explain why not: _____

• (if **no**, skip to **11c**) •

• I stopped simulating because I was extremely confident that there were no remaining timing errors in the design..... [ ] Yes (if **yes**, skip to **11c**) [ ] No

I thought that there could be remaining timing errors...	not at all accurate			somewhat accurate			very accurate	do not know
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.....	1	2	3	4	5	6	7	[ ]
• but the rate of timing error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining timing errors.....	1	2	3	4	5	6	7	[ ]
• but I felt that it would be easy to fix the remaining timing errors in the hardware prototype.....	1	2	3	4	5	6	7	[ ]
• Other: _____	1	2	3	4	5	6	7	[ ]

(11c) *About signal quality errors:*

• I used computer simulation to find signal quality errors before switching to the first hardware prototype.

[ ] Yes

[ ] No. Please explain why not: _____

• (if **no**, skip to **part 4**) •

• I stopped simulating because I was extremely confident that there were no remaining signal quality errors in the design.. [ ] Yes (if **yes**, skip to **part 4**) [ ] No

I thought that there could be remaining signal quality errors..	not at all accurate		somewhat accurate			very accurate		do not know
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.....	1	2	3	4	5	6	7	[ ]
• but the rate of signal quality error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining signal quality errors.....	1	2	3	4	5	6	7	[ ]
• but I felt that it would be easy to fix the remaining signal quality errors in the hardware prototype.....	1	2	3	4	5	6	7	[ ]
• other: _____	1	2	3	4	5	6	7	[ ]

PART 4: SOME INFORMATION ABOUT THE USE OF PROTOTYPES

This part should help us to understand the use of hardware prototypes in your last design discussed in part 3.

	never		sometimes			many times		do not know
(1) ***After finding errors in the hardware prototype,*** how often did you go back to computer simulation (e.g. for error analysis or verification of an error fix) ?.....	1	2	3	4	5	6	7	[ ]

(2) For the following question, a **prototype iteration** occurs whenever you make a change to any part of the design prototype and subsequently test it. (For example, reprogramming a single EPLD and retesting the design will count as one prototype iteration.)

• Please estimate the number of **prototype iterations** that were required before the design was complete:

Approximate number of iterations: _____ [ ] >25 [ ] Do not know

Percentage of iterations due to design errors: _____ % [ ] Do not know

Percentage of iterations due to specification changes: _____ % [ ] Do not know

Percentage of iterations due to other reasons: _____ % [ ] Do not know

(Please explain other reasons if applicable: _____)

(3) Please provide a rough estimate of the time it took to complete your *last* design.

• Total design time: _____ man-months. [ ] Do not know

Percentage of total design time spent on ***design specifying***: _____ % [ ] Do not know

Percentage of total design time spent on ***design development***: _____ % [ ] Do not know

Percentage of total design time spent on ***design verification***: _____ % [ ] Do not know

Percentage of total design time spent on ***design prototyping***: _____ % [ ] Do not know

Percentage of total design time spent on ***prototype evaluation***: _____ % [ ] Do not know

D.2 The ASIC Questionnaire

(Please turn to next page.)

(This area is intentionally left blank)

MASSACHUSETTS INSTITUTE OF TECHNOLOGY (M.I.T.)

DEPARTMENTS OF ELECTRICAL ENGINEERING AND MANAGEMENT

The M.I.T. ASIC Design Survey

Questionnaire Instructions

In completing this questionnaire, we are asking your cooperation to participate in an important national study designed to obtain a better understanding of design in general, and the design of ASICs (i.e. gate arrays, cell-based, full custom, but not programmable logic) in particular. You were carefully selected from a large pool of very qualified design engineers and it is of utmost importance to our research that you respond (a stamped return envelope is enclosed).

If you do not design ASICs, please pass the survey on to someone who does. If you cannot complete the questionnaire and you cannot pass it on to someone else, please write *"wrong experience"* on the questionnaire and send it back to us.

Interested in winning a MIT T-shirt? We will hold a lottery at which *three* designers will receive a MIT T-shirt (see section below).

Thank you for your cooperation !

Stefan Thomke, Research Assistant
Professor Eric von Hippel, Research Director

ENTRY SECTION FOR T-SHIRT LOTTERY

[ ] Yes, I would like to participate in the MIT T-shirt lottery. My size is: S M L XL
(circle one)

My name is: _____
(last name) (first name)

PART 1: YOUR BACKGROUND AND EXPERIENCE

This part will help us to learn about your experience.

(1) How long have you been a circuit designer? _____ years and _____ months.

(2) How long have you been a user of design simulation tools? _____ years and _____ months.

	No, not at all						Yes, absolutely
(3) When you design ASICs, would you describe your design style as incremental (i.e. design a little, test a little, and so on)?	1	2	3	4	5	6	7

PART 2: THE EFFECTIVENESS OF SIMULATION AND PROTOTYPING IN DEBUGGING ERRORS

In the following questions, you will be asked about the general use and effectiveness of **computer simulation** and **prototyping** (i.e. the fabricated ASIC prototype) in debugging errors. The following definitions will be used:

- ***Logic errors:*** errors due to faulty logic behavior of design (*examples*: reversed select lines of a decoder; negation of an input signal).
- ***Timing errors:*** errors due to faulty temporal behavior of design (*examples*: insufficient setup and hold times).
- ***Signal quality errors:*** errors due to degradation of signal quality during signal transmission (*example*: introduction of noise in high frequency applications due to excessive line proximity on printed circuit board).

Which is more effective: simulation or prototyping?	***simulation*** more effective			no difference		***prototype*** more effective		do not know
Questions about logic errors								
Which method is more effective in...								
• detecting all logic errors if you could run all possible test cases (i.e. which method is more accurate) ?	1	2	3	4	5	6	7	[ ]
• the analysis of logic errors and finding the error cause?	1	2	3	4	5	6	7	[ ]
• fixing logic errors (and verifying the fix)?	1	2	3	4	5	6	7	[ ]
Questions about timing errors								
Which method is more effective in...								
• detecting all timing errors if you could run all possible test cases (i.e. which method is more accurate)?	1	2	3	4	5	6	7	[ ]
• the analysis of timing errors and finding the error cause?	1	2	3	4	5	6	7	[ ]
• fixing timing errors (and verifying the fix)?	1	2	3	4	5	6	7	[ ]
Questions about signal quality errors								
Which method is more effective in...								
• detecting all signal quality errors if you could run all possible test cases (i.e. which method is more accurate)?	1	2	3	4	5	6	7	[ ]
• the analysis of signal quality errors and finding the error cause?	1	2	3	4	5	6	7	[ ]
• fixing signal quality errors (and verifying the fix)?	1	2	3	4	5	6	7	[ ]

PART 3: SWITCHING FROM COMPUTER SIMULATION TO THE FIRST PROTOTYPE

This part should help us to learn some specifics about the last ASIC design that you were involved in.

- Please think back to the ***last*** design:

  (a) that involved the design of an ASIC (e.g. gate arrays, cell-based, full custom)

  AND

  (b) during which you were involved in deciding <u>when</u> to switch from computer simulation to the <u>first</u> fabricated prototype.

  This design will be referred to as "your last design" from this point forward.

(1) Please indicate when your *last* design was completed: completed _____ months ago.

(2) Please indicate the type of ASIC you designed: [ ] Gate Array [ ] Full Custom
[ ] Cell-Based [ ] Others:_____

(3) Please name the ASIC manufacturer: _____

(4) Please estimate the <u>total</u> size of your *last* design: approx. _____gates (incl. copied cells)

(5) Please estimate the percentage of your design that was copied (e.g. repeated cells):_____% of total gates

(6) Please indicate the <u>total</u> number of ASICs that were or will be fabricated: approx. _____ ASICs

	completely analog						completely digital
(7) Please indicate if your *last* design was.....	1	2	3	4	5	6	7

(8) The following list contains general statements about your *last* design.

Please indicate the accuracy of each statement.	not at all accurate			somewhat accurate		very accurate		cannot answer
• I have done very similar designs before.....	1	2	3	4	5	6	7	[ ]
• The design's run speed was pushed to its absolute limit.....	1	2	3	4	5	6	7	[ ]
• The design had a high degree of logic complexity.....	1	2	3	4	5	6	7	[ ]
• Before starting to simulate, I knew <u>what type</u> of errors (i.e. logic, timing, and signal) the design would have <u>and</u> what their relative distribution would be.....	1	2	3	4	5	6	7	[ ]
• Before starting to simulate, I knew <u>how many</u> errors of each error type the design would have.....	1	2	3	4	5	6	7	[ ]

If you knew something about the type, the distribution and number of errors before starting simulation, please explain why: _____

_____

_____

(9) Did you simulate your design **on paper** *before* switching to computer simulation or prototyping?..... [ ] Yes [ ] No (if **no**, skip to **(10)**)

Please indicate the amount of paper simulations done...	very few						very many
• before schematic capture.....	1	2	3	4	5	6	7
• after schematic capture.....	1	2	3	4	5	6	7

(10) Think of **the time** you switched from **computer simulation to prototyping** (i.e. the first *fabricated* prototype). The following questions relate to your **reasons for stopping simulation** and prototyping the ASIC.

(10a) ***About logic errors :***

• I used computer simulation to find logic errors before switching to the first fabricated prototype.

[ ] **Yes**

[ ] **No.** Please explain why not: _____

• (if **no**, skip to **10b**) •

• I stopped simulating because I was extremely confident that there were no remaining logic errors in the design..... [ ] Yes (if **yes**, skip to **10b**) [ ] No

I thought that there could be remaining logic errors...	not at all accurate			somewhat accurate			very accurate	do not know
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.....	1	2	3	4	5	6	7	[ ]
• but the rate of logic error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining logic errors.....	1	2	3	4	5	6	7	[ ]
• but I felt that it would be easy to fix the remaining logic errors in the fabricated ASIC prototype.....	1	2	3	4	5	6	7	[ ]
• Other: _____	1	2	3	4	5	6	7	[ ]

(10b) ***About timing errors:***

• I used computer simulation to find timing errors before switching to the first fabricated prototype.

[ ] **Yes**

[ ] **No.** Please explain why not: _____

• (if **no**, skip to **10c**) •

• I stopped simulating because I was extremely confident that there were no remaining timing errors in the design..... [ ] Yes (if **yes**, skip to **10c**) [ ] No

I thought that there could be remaining timing errors...	not at all accurate			somewhat accurate			very accurate	do not know
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.....	1	2	3	4	5	6	7	[ ]
• but the rate of timing error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining timing errors.....	1	2	3	4	5	6	7	[ ]
• but I felt that it would be easy to fix the remaining timing errors in the fabricated ASIC prototype.....	1	2	3	4	5	6	7	[ ]
• Other: _____	1	2	3	4	5	6	7	[ ]

(10c) *About signal quality errors:*

- I used computer simulation to find signal quality errors before switching to the first fabricated prototype.

  [ ] Yes

  [ ] No. Please explain why not: _____

• (if **no**, skip to **part 4**) •

- I stopped simulating because I was extremely confident that there were no remaining signal quality errors in the design.. [ ] Yes (if **yes**, skip to **part 4**) [ ] No

I thought that there could be remaining signal quality errors..	not at all accurate			somewhat accurate			very accurate	do not know
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.....	1	2	3	4	5	6	7	[ ]
• but the rate of signal quality error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining signal quality errors.....	1	2	3	4	5	6	7	[ ]
• but I felt that it would be easy to fix the remaining signal quality errors in the fabricated ASIC prototype.....	1	2	3	4	5	6	7	[ ]
• Other: _____	1	2	3	4	5	6	7	[ ]

PART 4: SOME INFORMATION ABOUT EMULATION AND PROTOTYPING

This part will help us to understand the use of emulation and prototyping in your last design discussed in part 3.

(1) Did you use **hardware emulation** (for example, Zycad or Quickturn) before committing to the first ASIC prototype?..... [ ] Yes [ ] No (if no, skip to (2))

- How many times did you emulate your ASIC (for example, making a change to your design and re-emulating it counts as one emulation) ? _____ emulations

(2) For the following question, a **prototype iteration** occurs whenever you make a change to any part of the fabricated ASIC prototype and subsequently verify it. (For example, making a physical change to the ASIC prototype and retesting the design will count as one ASIC prototype iteration.)

- Please estimate the number of ASIC **prototype iterations** that were required before design completion:

  Approximate number of iterations: _____ iterations [ ] Do not know

  Percentage of iterations due to design errors: _____ % [ ] Do not know

  Percentage of iterations due to specification changes: _____ % [ ] Do not know

  Percentage of iterations due to other reasons: _____ % [ ] Do not know

  (Please explain other reasons if applicable: _____)

(3) Please provide a rough estimate of the time it took to complete your *last* design.

- Total design time: _____ man-months. [ ] Do not know

  Percentage of total design time spent on ***design specifying***: _____ % [ ] Do not know

  Percentage of total design time spent on ***design development***: _____ % [ ] Do not know

  Percentage of total design time spent on ***design verification***: _____ % [ ] Do not know

  Percentage of total design time spent on ***design prototyping***: _____ % [ ] Do not know

  Percentage of total design time spent on ***prototype evaluation***: _____% [ ] Do not know

Appendix E

Second Field Study: Summary of Survey Responses

On the following pages, you will find summary statistics of responses to the two questionnaires that were sent to 500 EPLD designers and 500 ASIC designers on August 12, 1994. The values shown are based on all responses (i.e. the raw data) found on returned questionnaires. Sample sizes smaller than the number of returned questionnaires are due to missing responses only. (Hypothesis tests in other chapters may use smaller sample sizes because some answers were considered invalid for testing purposes. In such cases, an explanation is provided.) For an explanation of the response scales, please consult appendix D.

E.1 EPLD Questionnaire Results

(Please turn to next page.)

PART 1: YOUR BACKGROUND AND EXPERIENCE

This part will help us to learn about your experience.	$\bar{x}$	s	n
(1) How long have you been a circuit designer (years)?	10.67	6.80	202
(2) How long have you been a user of design simulation tools (years)?	5.02	3.72	201
(3) When you design circuits containing EPLDs, would you describe your design style as incremental (i.e. design a little, test a little, and so on)?	4.18	1.76	203

PART 2: THE EFFECTIVENESS OF SIMULATION AND PROTOTYPING IN DEBUGGING ERRORS

In the following questions, you will be asked about the general use and effectiveness of **computer simulation** and **prototyping** in debugging errors in circuits containing EPLDs. The following definitions will be used:

- ***Logic errors:*** errors due to faulty logic behavior of design (*examples*: reversed select lines of a decoder; negation of an input signal).
- ***Timing errors:*** errors due to faulty temporal behavior of design (*examples*: insufficient setup and hold times).
- ***Signal quality errors:*** errors due to degradation of signal quality during signal transmission (*example*: introduction of noise in high frequency applications due to excessive line proximity on printed circuit board).

Which is more effective: simulation or prototyping?	$\bar{x}$	s	n
Questions about logic errors			
Which method is more effective in...			
• detecting all logic errors if you could run all possible test cases (i.e. which method is more accurate) ?	2.87	2.03	200
• the analysis of logic errors and finding the error cause?	2.32	1.47	200
• fixing logic errors (and verifying the fix)?	2.64	1.83	201
Questions about timing errors			
Which method is more effective in...			
• detecting all timing errors if you could run all possible test cases (i.e. which method is more accurate)?	3.44	2.07	199
• the analysis of timing errors and finding the error cause?	3.02	1.84	200
• fixing timing errors (and verifying the fix)?	3.62	2.09	200
Questions about signal quality errors			
Which method is more effective in...			
• detecting all signal quality errors if you could run all possible test cases (i.e. which method is more accurate)?	5.66	1.57	185
• the analysis of signal quality errors and finding the error cause?	5.40	1.62	187
• fixing signal quality errors (and verifying the fix)?	5.71	1.51	187

PART 3: SWITCHING FROM COMPUTER SIMULATION TO THE FIRST PROTOTYPE

This part should help us to learn some specifics about the last hardware design that you were involved in.

- Please think back to the ***last*** design:

  (a) that involved the usage of electrically-programmable logic devices (EPLDs) (e.g. FPGA, PLDs)

  AND

  (b) during which you were involved in deciding when to switch from computer simulation to the first hardware prototype.

  This design will be referred to as "your last design" from this point forward.

(1)	Please indicate when your *last* design was completed (months):	$\bar{x}$ = 6.89	s = 7.82	n = 200
(2)	Please indicate the number of EPLDs used in your *last* design:	= 6.93	s =16.11	n = 199
(3)	Please indicate the type of EPLDs used in your *last* design:	***(please contact author for details)***		
(4)	Please indicate the EPLD manufacturer:	***(please contact author for details)***		
(5)	Please estimate the total size of your *last* design:	$\bar{x}$ = 20.26 k	s = 81.42 k	n = 174
(6)	Please estimate the percentage of your design that was copied:	$\bar{x}$ = 23.01%	s = 21.54%	n = 193
(7)	Please indicate the total number of boards ... fabricated:	$\bar{x}$ =1909.36	s = 9023.94	n = 198
(8)	Please indicate if your *last* design was (analog=1; digital=7)	$\bar{x}$ = 5.68	s = 1.32	n = 203

(9) The following list contains general statements about your *last* design.

Please indicate the accuracy of each statement.	$\bar{x}$	s	n
• I have done very similar designs before.....	4.22	1.81	202
• The design's run speed was pushed to its absolute limit.....	3.53	2.06	201
• The design had a high degree of logic complexity.....	4.39	1.67	203
• Before starting to simulate, I knew what type of errors (i.e. logic, timing, and signal) the design would have and what their relative distribution would be.....	3.12	1.65	197
• Before starting to simulate, I knew how many errors of each error type the design would have.....	2.03	1.18	193

If you knew something about the type, the distribution and number of errors before starting simulation, please explain why: *Typical answers:* knew critical areas (some designers described them); experience; have done similar designs before; knew when they were pushing the limit; mental calculations prior to simulation.

(10) Did you simulate your design **on paper** before switching to computer simulation or prototyping? ..... *Yes*: 96 *No*: 107 $n = 203$

Please indicate the amount of paper simulations done...	$\bar{x}$	s	n
• before schematic capture	2.87	1.50	95
• after schematic capture	2.71	1.71	95

(11) Think of **the first time** you decided to switch from **computer simulation to prototyping** (i.e. *the first prototype*). The following questions relate to your **reasons for stopping simulation** and switching to prototyping.

(11a) ***About logic errors:***

• I used computer simulation to find logic errors before switching to the first hardware prototype. *Yes*: 180 *No*: 17 $n = 197$

• I stopped simulating because I was extremely confident that there were no remaining logic errors in the design *Yes*: 91 *No*: 87 $n = 178$

I thought that there could be remaining logic errors..	$\bar{x}$	s	n
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation	2.96	1.78	83
• but the rate of logic error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining logic errors	4.85	1.80	84
• but I felt that it would be easy to fix the remaining logic errors in the hardware prototype	4.97	1.74	85
• Other: ***Typical answer: project schedule forced me to stop***	6.47	1.01	17

(11b) ***About timing errors:***

• I used computer simulation to find timing errors before switching to the first hardware prototype. *Yes*: 132 *No*: 65 $n = 197$

• I stopped simulating because I was extremely confident that there were no remaining timing errors in the design *Yes*: 59 *No*: 71 $n = 130$

I thought that there could be remaining timing errors...	$\bar{x}$	s	n
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation	2.92	1.81	65
• but the rate of timing error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining timing errors	4.70	1.84	67
• but I felt that it would be easy to fix the remaining timing errors in the hardware prototype	4.39	1.99	65
• Other: ***Typical answer: project schedule forced me to stop***	6.64	0.51	11

(11c) *About signal quality errors:*

- I used computer simulation to find signal quality errors before switching to the first hardware prototype.

	Yes: 34	*No*: 163	$n = 197$

- I stopped simulating because I was extremely confident that there were no remaining signal quality errors in the design

	Yes: 17	*No*: 15	$n = 32$

I thought that there could be remaining signal quality errors..	$\bar{x}$	s	n
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation	2.93	1.77	14
• but the rate of signal quality error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining signal quality errors.....	4.57	2.10	14
• but I felt that it would be easy to fix the remaining signal quality errors in the hardware prototype.....	4.00	2.20	15
• Other: ***Typical answer:*** **project schedule forced me to stop**	6.50	0.71	2

PART 4: SOME INFORMATION ABOUT THE USE OF PROTOTYPES

This part should help us to understand the use of hardware prototypes in your last design discussed in part 3.

(1) *After finding errors in the hardware prototype*, how often did you go back to computer simulation (e.g. for error analysis or verification of an error fix) ?	$\bar{x}$	s	n
	4.28	2.01	201

(2) For the following question, a **prototype iteration** occurs whenever you make a change to any part of the design prototype and subsequently test it. (For example, reprogramming a single EPLD and retesting the design will count as one prototype iteration.)

- Please estimate the number of **prototype iterations** that were required before the design was complete:

	$\bar{x}$	s	n
Approximate number of iterations:	12.37	15.39	193
Percentage of iterations due to design errors:	52.69%	31.81%	178
Percentage of iterations due to specification changes:	29.24%	28.30%	176
Percentage of iterations due to other reasons:	18.03%	24.20%	176

(3) Please provide a rough estimate of the time it took to complete your *last* design.

	$\bar{x}$	s	n
• Total design time (man-months):	7.09	7.47	193
Percentage of total design time spent on ***design specifying***:	17.59%	12.66%	188
Percentage of total design time spent on ***design development***:	32.27%	15.15%	190
Percentage of total design time spent on ***design verification***:	17.14%	8.96%	190
Percentage of total design time spent on ***design prototyping***:	15.79%	10.05%	189
Percentage of total design time spent on ***prototype evaluation***:	17.94%	11.54%	184

E.2 The ASIC Questionnaire Results

(Please turn to next page.)

(This area is intentionally left blank)

PART 1: YOUR BACKGROUND AND EXPERIENCE

This part will help us to learn about your experience.	$\bar{x}$	s	n
(1) How long have you been a circuit designer?	10.10	6.07	186
(2) How long have you been a user of design simulation tools?	7.74	4.21	188
(3) When you design ASICs, would you describe your design style as incremental (i.e. design a little, test a little, and so on)?	4.90	1.65	187

PART 2: THE EFFECTIVENESS OF SIMULATION AND PROTOTYPING IN DEBUGGING ERRORS

In the following questions, you will be asked about the general use and effectiveness of **computer simulation** and **prototyping** (i.e. the fabricated ASIC prototype) in debugging errors. The following definitions will be used:

- ***Logic errors:*** errors due to faulty logic behavior of design (*examples*: reversed select lines of a decoder; negation of an input signal).
- ***Timing errors:*** errors due to faulty temporal behavior of design (*examples*: insufficient setup and hold times).
- ***Signal quality errors:*** errors due to degradation of signal quality during signal transmission (*example*: introduction of noise in high frequency applications due to excessive line proximity on printed circuit board).

Which is more effective: simulation or prototyping?	$\bar{x}$	s	n
Questions about logic errors			
Which method is more effective in...			
• detecting all logic errors if you could run all possible test cases (i.e. which method is more accurate) ?	2.56	1.79	187
• the analysis of logic errors and finding the error cause?	1.65	1.17	187
• fixing logic errors (and verifying the fix)?	1.81	1.35	187
Questions about timing errors			
Which method is more effective in...			
• detecting all timing errors if you could run all possible test cases (i.e. which method is more accurate)?	3.01	2.13	184
• the analysis of timing errors and finding the error cause?	2.16	1.49	185
• fixing timing errors (and verifying the fix)?	2.36	1.76	185
Questions about signal quality errors			
Which method is more effective in...			
• detecting all signal quality errors if you could run all possible test cases (i.e. which method is more accurate)?	5.28	1.91	172
• the analysis of signal quality errors and finding the error cause?	4.55	1.98	172
• fixing signal quality errors (and verifying the fix)?	4.73	1.93	172

PART 3: SWITCHING FROM COMPUTER SIMULATION TO THE FIRST PROTOTYPE

This part should help us to learn some specifics about the last ASIC design that you were involved in.

- Please think back to the ***last*** design:

  (a) that involved the design of an ASIC (e.g. gate arrays, cell-based, full custom)

  AND

  (b) during which you were involved in deciding <u>when</u> to switch from computer simulation to the <u>first</u> fabricated prototype.

  This design will be referred to as "your last design" from this point forward.

(1)	Please indicate when your *last* design was completed:	$\bar{x}$ = 11.30	s = 14.00	n = 186
(2)	Please indicate the type of ASIC you designed:	***(please contact author for details)***		
(3)	Please name the ASIC manufacturer:	***(please contact author for details)***		
(4)	Please estimate the <u>total</u> size of your *last* design (in gates):	$\bar{x}$ = 95.44 k	s = 176.26 k	n = 171
(5)	Please estimate the percentage of your design that was copied:	$\bar{x}$ = 30.17%	s = 26.18%	n = 174
(6)	Please indicate the <u>total</u> number of ASICs ... fabricated:	$\bar{x}$ = 174.38 k	s = 418.22 k	n = 158
(7)	Please indicate if your *last* design was (analog=1; digital=7)	$\bar{x}$ = 5.85	s = 1.72	n = 187

(8) The following list contains general statements about your *last* design.

Please indicate the accuracy of each statement.	$\bar{x}$	s	n
• I have done very similar designs before.....	4.23	1.99	185
• The design's run speed was pushed to its absolute limit.....	4.45	2.07	186
• The design had a high degree of logic complexity.....	4.86	1.82	188
• Before starting to simulate, I knew <u>what type</u> of errors (i.e. logic, timing, and signal) the design would have <u>and</u> what their relative distribution would be.....	3.36	1.69	185
• Before starting to simulate, I knew <u>how many</u> errors of each error type the design would have.....	2.09	1.42	181

If you knew something about the type, the distribution and number of errors before starting simulation, please explain why: *<u>Typical answers:</u>* <u>knew critical areas (some designers described them); experience; have done similar designs before; knew when they were pushing the limit; mental calculations prior to simulation.</u>

(9) Did you simulate your design **on paper** *before* switching to computer simulation or prototyping?..... *Yes*: 58 *No*: 129 n = 187

Please indicate the amount of paper simulations done...	$\bar{x}$	s	n
• before schematic capture.....	3.09	1.54	58
• after schematic capture.....	2.46	1.57	56

(10) Think of **the time** you switched from **computer simulation to prototyping** (i.e. the first *fabricated* prototype). The following questions relate to your **reasons for stopping simulation** and prototyping the ASIC.

(10a) ***About logic errors :***

• I used computer simulation to find logic errors before switching to the first fabricated prototype.
Yes: 180 *No*: 7 n = 187

• I stopped simulating because I was extremely confident that there were no remaining logic errors in the design *Yes*: 105 *No*: 72 n = 177

I thought that there could be remaining logic errors...	$\bar{x}$	s	n
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.....	3.27	2.17	70
• but the rate of logic error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining logic errors.....	4.55	1.84	71
• but I felt that it would be easy to fix the remaining logic errors in the fabricated ASIC prototype.....	2.56	1.95	70
• Other: ***Typical answer:*** **project schedule forced me to stop**	6.59	0.63	29

(10b) ***About timing errors:***

• I used computer simulation to find timing errors before switching to the first fabricated prototype.
Yes: 173 *No*: 13 n = 186

• I stopped simulating because I was extremely confident that there were no remaining timing errors in the design *Yes*: 102 *No*: 64 n = 166

I thought that there could be remaining timing errors...	$\bar{x}$	s	n
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation.....	3.21	2.08	62
• but the rate of timing error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining timing errors.....	4.34	1.95	62
• but I felt that it would be easy to fix the remaining timing errors in the fabricated ASIC prototype.....	2.68	2.00	62
• Other: ***Typical answer:*** **project schedule forced me to stop**	6.67	0.57	24

(10c) ***About signal quality errors:***

- I used computer simulation to find signal quality errors before switching to the first fabricated prototype.

	Yes: 77	*No*: 106	$n = 183$
• I stopped simulating because I was extremely confident that there were no remaining signal quality errors in the design	*Yes*: 38	*No*: 36	$n = 74$

I thought that there could be remaining signal quality errors..	$\bar{x}$	s	n
• but I stopped because I had completed a set of test vectors that was preplanned prior to starting simulation	3.71	2.04	34
• but the rate of signal quality error detection became ineffectively low and I expected the prototype to be much faster at detecting the remaining signal quality errors.....	4.47	1.96	34
• but I felt that it would be easy to fix the remaining signal quality errors in the fabricated ASIC prototype.....	3.42	2.06	33
• Other: ***Typical answer: project schedule forced me to stop***	6.67	0.71	9

PART 4: SOME INFORMATION ABOUT EMULATION AND PROTOTYPING

This part will help us to understand the use of emulation and prototyping in your last design discussed in part 3.

(1) Did you use **hardware emulation** (for example, Zycad or Quickturn) before committing to the first ASIC prototype? ..... *Yes*: 37 *No*: 150 $n = 187$

- How many times did you emulate your ASIC (for example, making a change to your design and re-emulating it counts as one emulation)? $\bar{x} = 17.32$ $s = 25.67$ $n = 31$

(2) For the following question, a **prototype iteration** occurs whenever you make a change to any part of the fabricated ASIC prototype and subsequently verify it. (For example, making a physical change to the ASIC prototype and retesting the design will count as one ASIC prototype iteration.)

- Please estimate the number of ASIC **prototype iterations** that were required before design completion:

	$\bar{x}$	s	n
Approximate number of iterations:	1.85	1.94	151
Percentage of iterations due to design errors:	47.82%	38.20%	104
Percentage of iterations due to specification changes:	26.49%	33.15%	104
Percentage of iterations due to other reasons:	25.67%	37.48%	104

(3) Please provide a rough estimate of the time it took to complete your *last* design.

	$\bar{x}$	s	n
• Total design time (man-months):	45.00	126.53	160
Percentage of total design time spent on ***design specifying***:	17.10%	10.45%	153
Percentage of total design time spent on ***design development***:	28.77%	11.53%	154
Percentage of total design time spent on ***design verification***:	31.94%	13.89%	154
Percentage of total design time spent on ***design prototyping***:	10.07%	8.59%	141
Percentage of total design time spent on ***prototype evaluation***:	14.30%	9.98%	143

Bibliography

B.1 Literature Related to Engineering and Science

Amato, Ivan (1992) "Speeding Up a Chemical Game of Chance." Science (July 17, 1992): 330-1.

Basili, Victor and Richard Selby (1987) "Comparing the Effectiveness of Software Testing Strategies." IEEE Transactions on Software Engineering (December 1987): 1278-1295.

Bergman, Sven and John Gittens (1985) Statistical Methods for Pharmaceutical Research Planning. New York: M. Dekker, 1985.

Bhatia, Dinesh (1994) "Field Programmable Gate Arrays: A Cheaper Way of Customizing Product Prototypes." IEEE Potentials (February 1994): 16-19.

Boehm, Barry, Terence Gray, and Thomas Seewaldt (1984) "Prototyping versus Specifying: A Multiproject Experiment." IEEE Transactions on Software Engineering (May 1984): 290-302.

Boehm, Barry (1988) "A Spiral Model of Software Development and Enhancement." IEEE Computer (May 1988): 61-72.

Boehm, Barry (1981) Software Engineering Economics. Englewood Cliffs, NJ: Prentice Hall, 1981.

Carey, T. and R. Mason (1986) "Information System Prototyping: Techniques, Tools, and Methodologies." New Paradigms for Software Development, IEEE Computer Society, 1986.

Collofello, James and Scott Woodfield (1989) "Evaluating the Effectiveness of Reliability-Assurance Techniques." The Journal of Systems and Software (September 1989): 191-195.

Crinnion, John (1991) Evolutionary Systems Development. London: Pitman Publishing, 1991.

Dvorak, Paul and Leland Teschler (1993) "From Data to Finished Parts." Machine Design (October 1993): 75-80.

Einspruch, Norman, and Hilbert, Jeffrey (1991) Application Specific Integrated Circuit (ASIC) Technology. San Diego, CA: Academic Press, 1991.

Endres, Albert (1975) "An Analysis of Errors and Their Causes in System Programs." IEEE Transactions on Software Engineering (June 1975): 140-149.

Gell-Mann, Murray (1994) The Quark and the Jaguar. NY: W.H. Freeman and Company, 1994.

Glass, Robert (1993) "Error Detection: Which is Better, Reviews or Testing?" Journal of System Software 22 (1993): 1-2.

Gokhale, Maya, William Holmes, Andrew Kopser, Sara Lucas, Ronald Minnich, Douglas Sweely, and Daniel Lopresti (1991) "Building and Using a Highly Parallel Programmable Logic Array." Computer (January 1991):81-89.

Griffith, Mark (1993) "Rapid Prototyping Options Shrink Development Cost." Modern Plastics (September 1993): 45-47.

Hubka, Vladimir (1982) Principles of Engineering Design. London: Butterworth Scientific, 1982.

Jones, Alan (1992) "Design-to-Cost and Time-to-Market with FPGAs." Components in Electronics (January 1992): 6.

McCarthy, D. (1990) "Interpreting FPLD Gate-Density Data." High Performance Systems Programmable Logic Design Guide 6 (1990).

Middendorf, William (1986) Design of Devices and Systems. New York: Marcel Dekker, 1986.

Mullins, Mark (1990) Rapid Prototyping for Object-Oriented Systems. Reading, MA: Addison-Wesley, 1990.

Ostrofsky, Benjamin (1977) Design, Planning, and Development Methodology. Englewood Cliffs, New Jersey: Prentice-Hall, 1977.

Phadke, Madhav (1989) Quality Engineering Using Robust Design. Englewood Cliffs: Prentice Hall, 1989.

Selby, Richard (1990) "Empirically Based Analysis of Failures in Software Systems." IEEE Transactions on Reliability (October 1990): 444-454.

Shooman, Martin (1983) Software Engineering: Design, Reliability, and Measurement. New York: McGraw Hill, 1983.

Stryer, Lubert (1981) Biochemistry. New York: Freeman, 1981.

Suh, Nam (1990) The Principles of Design. New York: Oxford University Press, 1990.

Taguchi, Genichi (1987) System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Costs. White Plains, NY: Kraus International Publications, 1987.

Tallyn, Kent (1990) "Reprogrammable Missile: How an FPGA Adds Flexibility to the Navy's Tomahawk Missile." Design In (April 1990): 39-40.

Thayer, Thomas, Myron Lipow, and Eldred Nelson (1978) Software Reliability: A Study of Large Project Reality. New York: North-Holland Publishing Company, 1978.

Vopni, Ed (1991) "Overview of Programmable IC Architectures" Proceedings of the Fourth Annual IEEE International ASIC Conference and Exhibit, Rochester, NY, September 1991.

Waldrop, M. Mitchell (1993) Complexity - The Emerging Science at the Edge of Order and Chaos. NY: Simon & Schuster, 1993.

Walker, Rob (1992) Silicon Destiny: The History of Application Specific Integrated Circuits and LSI Logic Corporation. Milpitas, CA: C.M.C. Publications, 1992.

Wall, Matthew, Karl Ulrich, and Woodie Flowers (1991) "Evaluating Prototyping Technologies for Product Design." Working Paper No. 3334-91-MSA, Sloan School of Management, Massachusetts Institute of Technology, September 1991.

Weiss, David (1979) "Evaluating Software Development by Error Analysis: The Data from the Architecture Research Facility." The Journal of Systems and Software 1 (1979): 57-70.

Wohlers, Terry (1993) "Cashing in on Rapid Prototyping." Computer Aided Engineering (September 1993): 28-33.

Wright, T.P. (1936) "Factors Affecting the Cost of Airplanes." Journal of Aeronautical Science 3 (February 1936): 122-126.

B.2 Literature Related to Problem-Solving and Decision Theory

Adler, Paul and Kim Clark (1991) "Behind the Learning Curve: A Sketch of the Learning Process." Management Science 37 (March 1991): 267-281.

Alexander, Christopher (1964) Notes on the Synthesis of Form. Cambridge, MA: Harvard University Press, 1964.

Allen, Thomas J. (1966) "Studies of the Problem-Solving Process in Engineering Design." IEEE Transactions on Engineering Management EM-13, no.2: 72-83.

Barron, Jonathan (1988) Thinking and Deciding. New York: Cambridge University Press, 1988.

Bunn, Derek (1984) Applied Decision Analysis. New York: McGraw-Hill, 1984.

Clark, Kim, and Takahiro Fujimoto (1987) "Overlapping Problem Solving in Product Development." Harvard Business School, Working Paper#87-048, 1987.

Cohen, Wesley M., and Daniel A. Levinthal (1990) "Absorptive Capacity: A New Perspective on Learning and Innovation." Administrative Science Quarterly (March 1990): 128-52.

Collins, Harry M. (1990) Artficial Experts. Cambridge, MA: MIT Press, 1990.

Larkin, Jill, John McDermott, Dorothea Simon, and Herbert Simon (1980) "Expert and Novice Performance in Solving Physics Problems." Science 208 (June 1980): 1335-1342.

March, James (1994) A Primer on Decision-Making. New York: The Free Press, 1994.

Marples, David L. (1961) "The Decision of Engineering Design." IRE Transactions on Engineering Management 2 (June 1961): 55-71.

Newell, Allen and Herbert Simon (1972) Human Problem Solving. Englewood Cliffs, N.J.: Prentice-Hall, 1972.

Pople, Harry (1982) "Heuristic Methods for Imposing Structure on Ill-Structured Problems: The Structuring of Medical Diagnostics." in: Peter Szolovits (Editor), Artificial Intelligence in Medicine. Boulder, CO: Westview Press, 1982.

Popper, Karl (1972) Objective Knowledge: An Evolutionary Approach. London: Oxford University Press, 1972.

Reitman, W.R. (1965) Cognition and Thought. New York: Wiley, 1965.

Simon, Herbert (1955) "A Behavioral Model of Rational Choice." Quarterly Journal of Economics 69 (1955): 99-118.

Simon, Herbert (1976) "From Substantive to Procedural Rationality." in Method and Appraisal in Economics, edited by S.J. Latsis, Cambridge: Cambridge University Press, 1976.

Simon, Herbert (1978) "On How to Decide What to Do." The Bell Journal of Economics 9, no. 2 (1978): 494-507.

Simon, Herbert A. (1983) Models of Bounded Rationality: Behavioral Economics and Business Organization. Cambridge, MA: MIT Press, 1983.

Simon, Herbert A. (1981) The Sciences of the Artificial. Second Edition Cambridge, MA: MIT Press, 1969.

Simon, Herbert A. (1973) "The Structure of Ill-Structured Problems." Artificial Intelligence 4: 181-201.

Tyre, Marcie, and Eric von Hippel (1993) "Situated Trial and Error Learning in Organizations." Working Paper, Sloan School of Management, Massachusetts Institute of Technology, January 1993.

von Hippel, Eric (1994a) "'Sticky Information' and the Locus of Problem Solving: Implications for Innovation." Management Science 40, no. 4 (April 1994): 429-439.

von Hippel, Eric (1990) "Task Partitioning: An Innovation Variable." Research Policy 19, no. 5 (October 1990): 407-18.

von Hippel, Eric, and Marcie Tyre (1994b) "How 'Learning By Doing' Is Done: Problem Identification in Novel Process Equipment." Research Policy *forthcoming*.

B.3 Literature Related to Research Methods and Statistics

Box, George and Norman Draper (1987) Empirical Model-Building and Response Surfaces. New York: Wiley, 1987.

Box, George, William Hunter and Stuart Hunter (1978) Statistics for Experimenters. New York: Wiley, 1978.

Glaser, Barney and Anselm Strauss (1970) The Discovery of Grounded Theory: Strategies for Qualitative Research. Chicago: Aldine Publishing Company, 1970.

Hogg, Robert and Johannes Ledolter (1987) Engineering Statistics. New York: Macmillan, 1987.

Judd, Charles, Eliot Smith, and Loise Kidder (1991) Research Methods in Social Relations. Fort Worth, TX: Harcourt Brace Jovanovich, 1991.

Lockhart, Daniel (1984) Making Effective Use of Mailed Questionnaires. San Francisco: Jossey-Bass, 1984.

Miles, Matthew and Michael Huberman (1984) Qualitative Data Analysis. Beverly Hills, CA: Sage Publications, 1984.

Miller, Delbert (1977) Handbook of Research Design and Social Measurement. New York: McKay, 1977.

Montgomery, Douglas (1991) Design and Analysis of Experiments. New York: Wiley, 1991.

Ostle, Bernard and Linda Malone (1988) Statistic in Research. Ames: Iowa University Press, 1988.

Sachs, Lothar (1984) Applied Statistics: A Handbook of Techniques. New York: Springer-Verlag, 1984.

Winkler, Robert (1972) An Introduction to Bayesian Inference and Decision. New York: Holt, Reinhart and Winston, 1972.

B.4 Literature Related to Technology Management and Economics

Arrow, Kenneth (1984) The Economics of Information. Cambridge, MA: The Belknap Press of Harvard University Press.

Arrow, Kenneth (1962) "The Economic Implications of Learning by Doing." Review of Economic Studies 29 (1962): 155-173.

Chao, Linda (1993) Improving Quality and Time to Market in the VLSI Product Life Cycle. S.M. Thesis, Electrical Engineering and Management, Massachusetts Institute of Technology, June 1993.

Cesiel, Douglas (1993) A Structured Approach to Calibration Development for Automotive Diagnostic Systems. S.M. Thesis, Electrical Engineering and Management, Massachusetts Institute of Technology, May 1993.

Clark, Kim and Takahiro Fujimoto (1991) Product Development Performance. Boston, MA: Harvard Business School Press, 1991.

Clausing, Don (1993) Total Quality Development: A Step-by-Step Guide to World-Class Concurrent Engineering. New York: ASME Press, 1993.

Cooper, Kenneth (1993) "The Rework Cycle." Engineering Management Review (Fall 1993): 4-12.

Eppinger, Steven, Daniel Whitney, Robert Smith, and David Gebala (1993) "A Model-Based Method for Organizing Tasks in Product Development." Working Paper No. 3569-93-MS, Sloan School of Management, Massachusetts Institute of Technology, August 1990.

Feld, Bradley (1990) "The Changing Role of the User in the Development of Application Software." Working Paper, Sloan School of Management, Massachusetts Institute of Technology, August 1990.

Feigenbaum (1991) Total Quality Control. New York: McGraw Hill, 1991.

Hayes, Robert, Steven Wheelright, and Kim Clark (1988) Dynamic Manufacturing. New York: The Free Press, 1988.

Henderson, Rebecca (1993) "Flexible Integration as Core Competence: Architectural Innovation in Cardiovascular Drug Development." Mimeo, Stanford University, January 1993.

Keen, Peter (1991) Every Manager's Guide to Information Technology. Boston, MA: Harvard Business School Press, 1991.

Krishnan, Viswanathan, Steven Eppinger, and Daniel Whitney (1993) "A Model-Based Framework to Overlap Product Development Activities."

Working Paper#3635-93-MSA, Sloan School of Management, Massachusetts Institute of Technology, November 1993.

Nelson, Richard, and Sidney Winter (1982) An Evolutionary Theory of Economic Change. Cambridge, MA.: Harvard University Press, 1982.

Pisano, Gary (1994) "Knowledge, Integration, and the Locus of Learning: An Empirical Analysis of Process Development." Working Paper No. 95-006, Harvard Business School, August 1994.

Pralad, C.K. and Gary Hamel (1990) "The Core Competence of the Corporation." The Harvard Business Review (May-June 1990).

Reinertsen, Donald (1992) "The Mythology of Speed." Machine Design (March 26, 1992).

Reinertsen, Donald (1983) "Whodunit? The Search for New-Product Killers." Electronic Business (July 1983): 62-66.

Rosenberg, Nathan (1982) Inside the Black Box: Technology and Economics. New York: Cambridge University Press, 1982.

Senge, Peter, and Colleen Lannon (1990) "Managerial Microworlds." Technology Review (July 1990): 63-68.

Smith, Preston, and Donald Reinertsen (1991) Developing Products in Half the Time. New York: Van Nostrand Reinhold, 1991.

Smith, Robert, and Steven Eppinger (1991) "A Prediction Model of Sequential Iteration in Engineering Design." Working Paper No. 3160-90-MS, Sloan School of Management, Massachusetts Institute of Technology, November 1991.

Smith, Robert, and Steven Eppinger (1992) "Identifying Controlling Features of Engineering Design Iterations." Working Paper No. 3348-91-MS, Sloan School of Management, Massachusetts Institute of Technology, September 1992.

Steward, D.V. (1981) "The Design Structure System: A Method for Managing the Design of Complex Systems." IEEE Transactions on Engineering Management (August 1981): 71-74.

Ulrich, Karl and Steven Eppinger (1994) Product Design and Development. New York: McGraw-Hill, 1994.

von Hippel, Eric (1992) "Adapting Market Research to the Rapid Evolution of Needs for New Products and Services." Working Paper No. 3374-92-BPS, Sloan School of Management, Massachusetts Institute of Technology, January 1992.

von Hippel, Eric (1988) The Sources of Innovation. New York: Oxford University Press, 1988.

Wheelright, Steven, and Clark, Kim (1992) Revolutionizing Product Development. New York: The Free Press, 1992.