

AN ARCHITECTURE FOR HIGH-PERFORMANCE
SINGLE-CHIP VLSI TESTERS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

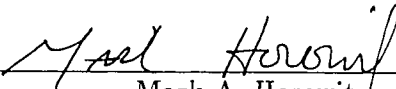
By

James A. Gasbarro

May 1989

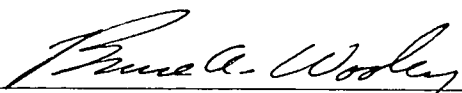
© Copyright 1989 by James A. Gasbarro
All Rights Reserved

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



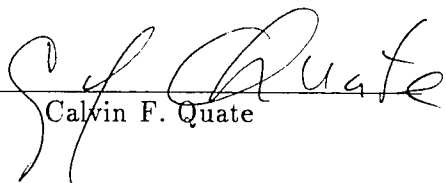
Mark A. Horowitz
(Principal Adviser)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



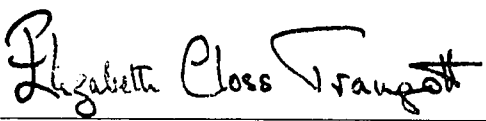
Bruce A. Wooley

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



Calvin F. Quate

Approved for the University Committee on Graduate Studies:



Dean of Graduate Studies

Abstract

Testing is an important factor in the production of useable custom integrated circuits. Verification of the functional and AC parametric characteristics of a device are usually performed on large and expensive test systems. This thesis presents a new approach to tester architecture that seeks to greatly reduce both the size and cost of these systems. The principal idea is to base the tester design on the same high-density technology as that of the devices being tested. Through the use of novel test vector compression techniques and closed-loop timing calibration methods, high performance and high density can be achieved in a CMOS technology. The proof is the implementation of a single-chip multi-channel tester which has the size and cost attributes of the very low-end testers, yet implements many of the features found on only the most expensive machines.

The high level of integration achieved results in a number of other advantages as well. The close proximity of the tester to the test device eliminates most of the signal transmission and loading issues encountered in larger systems. The extremely compact size enables in-circuit probing and performance analysis of the test device without custom fixturing. Finally, and perhaps most importantly, by implementing the tester in the same technology as that of the device to be tested, future upgrades of the tester capability can evolve along with the capabilities of the subject device.

Acknowledgments

My route through the Ph.D. program at Stanford was somewhat unusual since I was a full time employee at the Xerox Palo Alto Research Center during the entire course of my studies. As a result, I wish to thank many people for helping me finish successfully. Bob Sproull, Brian Reid, and Chuck Geschke first persuaded me to leave my previous job and come to Xerox with the lure that I could pursue my degree while working full time. Through the continued support of my managers at PARC, I was able to finally complete this work. I would particularly like to thank Bill Spencer, Ron Rider, Bob Ritchie, Mark Weiser, and Bryan Preas for their encouragement and backing.

The original idea for this thesis came from a conversation I had with Chuck Thacker nearly six years ago. I was new to PARC and wanted to learn about VLSI design. Chuck suggested that I build a "chip tester chip" as a quick introductory project. Sorry it took so long, Chuck. Rick Barth worked with me on that initial project and has been a continuing source of good ideas ever since. Don Curry and Louis Monier provided help with the design automation system to get the Testarossa chip assembled. Additional thanks to Rick and Louis for reading early drafts of this thesis. I am also indebted to Dan Greene and John Gill who taught me all I know about data compression, and Stafan Pfab and Gerhard Greisle, visiting scientists from Seimens, who provided assistance with analysis of compression algorithms.

I would also like to thank my advisor, Mark Horowitz, and the members of my

committee, Cal Quate and Bruce Wooley, for help in refining my final draft. None of this would have been possible without Mark's guidance and cooperation, particularly since I was not really a full-time student. I have greatly enjoyed and benefited from working with him.

Finally, I would like to thank my parents and Beth who put up with me, and encouraged me, when it seemed like the job was insurmountable.

Contents

| | |
|---|-----------|
| Abstract | iv |
| Acknowledgments | v |
| 1 Introduction | 1 |
| 1.1 Organization | 3 |
| 2 Background | 5 |
| 2.1 Introduction | 5 |
| 2.2 Pin Electronics | 6 |
| 2.3 Data Generator | 9 |
| 2.4 Limits of Current Systems | 12 |
| 3 Integrated Testers | 15 |
| 3.1 Introduction | 15 |
| 3.2 Don't Think Big | 16 |
| 3.3 Think Small | 16 |
| 3.4 Limits | 22 |
| 3.5 Summary | 24 |
| 4 Pin Electronics | 25 |
| 4.1 Introduction | 25 |

| | | |
|----------|---|-----------|
| 4.2 | Force Section | 26 |
| 4.3 | Formatter | 34 |
| 4.4 | Acquire Section | 39 |
| 4.5 | Calibration | 41 |
| 4.6 | Results and Summary | 45 |
| 5 | Vector Storage | 47 |
| 5.1 | Introduction | 47 |
| 5.2 | Compression Alternatives | 49 |
| 5.3 | Compression Requirements | 54 |
| 5.3.1 | Architectural implications | 55 |
| 5.3.2 | Bandwidth considerations | 56 |
| 5.3.3 | Compression efficiency | 56 |
| 5.4 | Model Selection | 58 |
| 5.5 | Decompressor Architectures | 59 |
| 5.5.1 | Huffman decompressor | 59 |
| 5.5.2 | Fiala-Greene decompressor | 62 |
| 5.6 | Compression Efficiencies | 65 |
| 5.7 | Analysis of Results | 71 |
| 5.7.1 | Architectural comparison | 71 |
| 5.7.2 | Bandwidth comparison | 74 |
| 5.7.3 | Compression efficiency comparison | 74 |
| 5.8 | Conclusions | 75 |
| 6 | Implementation | 77 |
| 6.1 | Introduction | 77 |
| 6.2 | System Overview | 77 |
| 6.3 | Testarossa Architecture | 80 |

| | | |
|----------|----------------------------------|------------|
| 6.3.1 | Pinout | 87 |
| 6.3.2 | Register model | 89 |
| 6.3.3 | Programmer's interface | 91 |
| 6.4 | Technology Constraints | 94 |
| 6.5 | Assembly | 94 |
| 6.6 | Results | 101 |
| 7 | Conclusions | 103 |
| 7.1 | Summary | 103 |
| 7.2 | Future Work | 104 |
| | Bibliography | 107 |

List of Tables

| | | |
|-----|--|-----|
| 4.1 | Delay element ranges and resolutions | 33 |
| 5.1 | Static Huffman coding | 66 |
| 5.2 | Copy command parameters | 67 |
| 5.3 | FG experimental results | 70 |
| 5.4 | Number of optimum efficiency occurrences vs. buffer size | 70 |
| 7.1 | Comparison of static and dynamic RAM implementations | 105 |

List of Figures

| | | |
|------|--|----|
| 4.1 | Pin electronics block diagram | 26 |
| 4.2 | Force timing generator | 27 |
| 4.3 | Edge deviation | 29 |
| 4.4 | Timing chain | 30 |
| 4.5 | Edge adjust (transistor widths in microns) | 30 |
| 4.6 | Shift register | 31 |
| 4.7 | Replica of an oscilloscope photo showing the edge resolution waveform (1 ns/div.) | 32 |
| 4.8 | Edge adjust unit with improved range | 32 |
| 4.9 | Schematic of format generator | 34 |
| 4.10 | Replica of an oscilloscope photo of the PE chip generating five output formats at 20 MHz (10 ns/div.) | 35 |
| 4.11 | Non-overlapped clock example | 36 |
| 4.12 | Addition of a fourth delay generator for calibrating RC mode | 37 |
| 4.13 | Pin driver circuitry | 38 |
| 4.14 | Acquisition and synchronization logic | 40 |
| 4.15 | Clocked analog voltage comparator | 40 |
| 4.16 | Phase detector | 42 |
| 4.17 | Schematic of D-type flip-flop | 44 |

| | | |
|------|--|----|
| 5.1 | Data compression through mapping | 49 |
| 5.2 | Taxonomy of compression alternatives | 50 |
| 5.3 | Huffman decoder block diagram | 60 |
| 5.4 | Codeword CAM cell schematic | 61 |
| 5.5 | Fiala-Greene decoder block diagram | 64 |
| 5.6 | FG compression vs. buffer size: FP and EU | 68 |
| 5.7 | FG compression vs. buffer size: DC | 69 |
| 5.8 | Branch operation | 72 |
| 6.1 | Single board integrated test system | 78 |
| 6.2 | Skew elimination by counter propagating signals | 80 |
| 6.3 | Block diagram of Testarossa chip | 81 |
| 6.4 | Control map | 82 |
| 6.5 | Test vector fields | 83 |
| 6.6 | Compression opcodes | 83 |
| 6.7 | Decompressor data path with bus widths | 85 |
| 6.8 | Error buffer | 86 |
| 6.9 | Pin electronics registers | 90 |
| 6.10 | Decompressor control registers | 92 |
| 6.11 | Testarossa floor plan | 95 |
| 6.12 | Testarossa layout | 96 |
| 6.13 | Typical schematic entry for standard cell control logic | 97 |
| 6.14 | Graphical FSM description implemented in standard cell logic | 97 |
| 6.15 | Decompressor data path segment | 99 |

Chapter 1

Introduction

Anyone who has designed a custom or semi-custom integrated circuit knows the excitement of producing the initial layout for a new device. After months of logic and mask design work, there is finally a picture that can be hung on the wall. Experienced designers know that this is only a fraction of the overall effort. Extensive simulation must be performed before the device can be submitted for fabrication. Eventually, the mask tape is shipped to the fabrication house and in a few weeks the actual silicon is in hand. But then what? Much more work remains to ensure that the chip performs the intended function. If the device is a high-volume part, then considerable resources can be brought to bear, typically in the form of a multi-million dollar test system devoted exclusively to that one device. In the case of the much lower volume Application Specific Integrated Circuit (ASIC) however, it is difficult to justify such expenditures, even though testing is equally important to the final application of the device. The result is that the testing is either contracted out to a vendor, resulting in the loss of direct access and control of the tester, or else performed on a somewhat inferior in-house test set-up. The low cost testers which are available for such purposes lack the speed, accuracy, and flexibility of their larger counterparts thus compromising the thoroughness of the testing, as well as requiring extensive custom

fixturing for each device to be tested. As bad as the situation is presently, it is likely to worsen as ASIC development costs continue to drop while performance increases. The result will be widening disparity between design and testing costs. The fundamental problem is that the current generation of testers are based on speed-aggressive technologies, such as Emitter Coupled Logic, to achieve the performance required to test ASICs. Due to thermal and device limitations, as well as market forces, such bipolar logic families are not experiencing the same growth in terms of integration density and speed as their MOS counterparts. The result is that it is difficult to track the performance improvement of the device under test by simply scaling the devices used in the tester.

This thesis focuses on a new approach to tester architectures. The principal idea is to base the tester design on the same technology as that of the devices to be tested. Ordinarily this would seem difficult, if not impossible, since one would not expect a system as complex as a tester to have performance characteristics superior to any other device fabricated in the same technology. A conclusion of this thesis, however, is that this goal is achievable through novel architecture and circuit design. The proof is a single-chip, multi-channel tester which has the size and cost attributes of the very low-end testers, yet implements many of the features found on only the most expensive machines. The high level of integration achieved results in a number of other advantages as well: the close proximity of the tester to the test device eliminates most of the signal transmission and loading issues encountered in larger systems; the extremely compact size enables in-circuit probing and performance analysis of the test device without custom fixturing; finally, and perhaps most importantly, by implementing the tester in the same technology as that of the device to be tested, future upgrades of the tester capability can evolve along with the capabilities of the subject devices.

1.1 Organization

The following chapter describes earlier developments in tester architectures. The basic structure of existing test systems is outlined and the significant developments in tester architectures are discussed. Future trends in circuit size and performance are presented along with the implications that they will have on tester performance requirements.

In Chapter 3 the basic properties of each of the components of large test systems are investigated with a view toward how to improve them using VLSI techniques. A new architecture is then proposed, with some discussion of the expected performance characteristics. The fundamental limits of this approach are then examined.

Chapters 4 and 5 focus specifically on the problems associated with the design of the two major functional elements of the system, the pin electronics and vector storage sub-systems. The pin electronics provides the formatting, timing, and sampling functions for the tester. Chapter 4 presents the detailed design of circuitry for achieving sub-nanosecond resolution in a VLSI technology. The techniques are demonstrated in an experimental four-channel pin electronics chip that was fabricated and tested to validate these ideas and verify performance data. Chapter 5 deals with the difficulty of storing a significant number of test vectors in a single-chip tester along with all of the other circuitry required in the device. A survey of data compression techniques examines possible means for increasing vector storage by storing compressed vectors and decompressing them in real-time. Specific requirements are outlined and the suitability of the various compression techniques to the vector storage problem is investigated.

The overall design and implementation of a single-chip tester device is presented in Chapter 6, including an overview of how the chip would be used to build a complete test system. The details of the actual device are presented along with performance and

programming data. The design constraints and assembly techniques for the device are also discussed.

Finally, Chapter 7 provides an overview of the results achieved and presents directions for future work.

Chapter 2

Background

2.1 Introduction

Since the early 1960's when the first integrated circuits were produced, a large number of different test systems have been developed. The earliest testers were fairly simple descendants of board-level testers. It was not long, however, before differences in board and integrated circuit testing caused the two types of testers to take on separate identities. Diversification in IC product lines resulted in further subdivision of the integrated circuit tester field into specific product categories. General purpose digital testers were developed for the growing SSI (Small Scale Integration) and MSI (Medium Scale Integration) market. Memory testers were devised with special purpose pattern generators to meet the needs of testing array structures. More recently, mixed analog/digital testers have appeared for testing a variety of products that combine linear and/or high voltage circuits with digital circuits on the same die. Each of these market segments faces its own challenges: general-purpose digital testers must track the test requirements of increasingly complex VLSI devices; memory testers must keep pace with the increasing speed and density requirements of new RAM technologies; and the mixed analog/digital testers must improve in their I/O flexibility

and signal analysis capabilities. The general-purpose digital field though, is in many ways the most challenging. Here, nearly all aspects of the tester architecture are changing at exponential rates. Ten years ago, the state-of-the-art in LSI (Large Scale Integration) technology was 50K transistors/chip, clock speeds of around 5 MHz, and pin counts which rarely exceeded 64. By the year 2000, as digital IC technology matures from VLSI (Very Large Scale Integration) to ULSI (Ultra Large Scale Integration), the number of transistors on a chip will exceed 100 million, new packaging technology will push pin counts into the thousands, and shrinking geometries will allow devices to operate at frequencies well in excess of 100 MHz [Me84, Bi83, KM78]. Coupled with these increases is the fact that these devices will encompass the memory and analog domains as well. It is not at all uncommon even today to find large embedded memories or analog I/Os in devices which are primarily digital. These demands of complexity, speed, pin-count, and I/O flexibility conspire to make the problem of building general-purpose testers quite formidable.

Tester architectures can be divided into two major components, the data generator and the pin electronics. The data generator produces the digital stimulus vectors for the *Device Under Test* (DUT), while the pin electronics is responsible for formatting these vectors to produce a waveform of the desired shape and timing, and for sampling the DUT outputs at the desired time.

2.2 Pin Electronics

In most test systems, there is a separate printed circuit card for every one or two pin electronics channels. In a tester with several hundred pins, it is easy to see why the pin electronics portion of a tester represents a significant fraction of the overall system cost. In many tester systems, resources, such as timing generators and voltage sources, are shared among pins, which restricts the flexibility of the tester but lowers its cost

[He83]. Another cost-reducing technique is to provide individual input and output pin types rather than a single general I/O pin. Testers of this class also frequently provide hardwired signal groups where all pins of the group have the same timing characteristics, in order to reduce the number of timing generators in the system. In such systems, it is up to the user to manually wire signals from the I/O groups to the proper DUT pins [IM85]. Even in medium range machines, the most expensive resources, such as timing generators, are often shared among pins, requiring complex switching matrices to distribute clock edges. Machines of this nature are said to have a *shared resource* architecture. At the high-end of the tester spectrum are machines which provide pin electronics with a general I/O architecture that has little or no sharing of resources [Bi83]. Since each pin is equivalent in power to every other pin, the organization is known as a *tester-per-pin*, or simply a *per-pin* architecture.

The term *pin electronics* has traditionally referred only to that portion of the tester system which is pin specific. In shared resource systems, this implies three major sub-circuits: the output driver, the input comparator, and a loading device. The output driver determines the output high (V_{OH}) and output low (V_{OL}) drive levels of the tester, while the input comparator determines the tester input sample threshold. Some testers have multiple input comparators which can be used either for measuring the V_{OH} and V_{OL} of the DUT or for measuring its output risetimes. A programmable current load device is sometimes included in the pin electronics for simulating the effects of a bipolar load on an output of the test device [Fe78].

In a per-pin architecture, pin electronics takes on a broader meaning. In such a system, the functions of edge timing generation and formatting are encompassed in the pin electronics in addition to drive, sense, and loading functions. Formatting is a means for obtaining higher I/O bandwidth to the DUT and also for reducing the size of the high speed memory necessary to hold the test vectors. The format unit takes the drive level supplied by the test vector and combines it with the output of

one or more timing generators to produce a pulse waveform of the desired shape. The position and duration of the pulse within the test cycle are controlled by the outputs of individual timing generators. Some of the standard format modes are Non-Return to Zero (NRZ), Return to Zero (RZ), Return to One (RO), and Return to Tri-State (RT).

The timing generators provide a clock edge at the desired point during the tester cycle. The quality of the placement of these edges is an important aspect of the tester performance, particularly in production oriented testers. Edge quality is measured in terms of two parameters: accuracy and skew. According to Abbot's definitions [Ab84]:

“Accuracy is the closeness by which measured change in the timing of a function edge corresponds to its programmed change.”

“Skew is the error between any two tester functions [edges] at the same programmed time, as perceived at the DUT.”

Thus, accuracy is measured in terms of the ability of the timing generator to place an edge at an absolute point in time, whereas skew is a relative measure between pins. When testing a device, the worst case error caused by accuracy and skew problems must be factored into every timing parameter that is to be measured. This technique, referred to as guardbanding, prevents bad devices from being accepted as good ones by the system due to poor measurements. The downside, however, is that guardbanding can also cause components which are actually good to fail the test operation, thus reducing production yield.

In order to maintain a high degree of accuracy and minimize skew, most testers have some built-in calibration mechanism. In low cost testers, this usually consists of “torque adjustments,” variable capacitors or delay lines which are adjusted manually at the factory or by a field technician. More expensive testers have built-in

auto-calibration facilities using fairly sophisticated techniques such as Time Domain Reflectometry (TDR) to measure signal delays right at the end of the probe tip. Typically, the calibration operation consists of two steps. First, the accuracy of each timing generator is calibrated to an external standard, usually a National Bureau of Standards traceable crystal. Second, the pin-to-pin skews are calibrated by switching a TDR system into the path to each probe tip and varying a per-pin skew adjustment [De83]. In a shared-resource architecture, the per-pin skew adjustment is dependent on the machine configuration specified by the test program. It therefore must be performed for each different device to be tested. In per-pin testers, the accuracy and skew adjustment can be combined into a single element, thereby reducing the number of skew and jitter-inducing delay blocks [CF83]. Other calibration techniques exist [Da87, SM87], but are minor variations on the same theme. Even with such sophisticated systems, the best overall system accuracies attained are on the order of one to two nanoseconds, which is rather poor considering the typical 50 to 100 ps inherent resolution. The reason for this poor system performance is primarily due to the large number of gate delays and physical components between the timing generators and the DUT. Gate counts as high as 50 are not uncommon, nor are tens of feet of interconnect path [CF83]. Each such element introduces drift and jitter terms which limit the maximum attainable accuracy. A solution to this problem is to integrate the pin electronics circuitry as much as possible. By nature of their small size and uniform thermal characteristics, integrated circuits are an ideal means for reducing component-induced errors [CR81].

2.3 Data Generator

The concept of the data generator for a high-speed tester seems simple enough: a data generator is a device which produces a single digital test vector that specifies

the drive and response data for each pin during every tester cycle. The difficulty comes when size and speed constraints are placed on the system. Most testers store three to four bits per pin in the vector memory to specify output drive level, expected input level, whether the pin is input or output, and whether or not to signal an error if the input compare operation fails. A state-of-the-art tester with 512 pins would therefore require a memory word width of 2048 bits. To test the most complex VLSI devices, the tester should be capable of delivering test sequences as long as one million vectors. Furthermore, if the tester is to cycle at speeds up to 100 MHz, high-speed (but low-density) ECL or GaAs RAMS would be necessary to supply the required bandwidth. It is easy to see how a data generator can stretch the limits of available memory technology. Many approaches have been taken to alleviate the constraints on this problem, all aimed at reducing the required amount of high-speed memory.

Schemes that have been implemented for reducing the memory size requirement for data generators fall into four main classifications: reduced functionality, memory overlay, algorithmic generation, and general compression. Some low-end testers reduce data generator word width by simply eliminating functionality. Pins of the tester are made to be either fixed-direction or unmaskable in order to reduce the number of high-speed memory chips. Such testers represent the least elegant approach to the problem since they reduce the potential tester applications.

The memory overlay schemes, which are somewhat more general, employ a small amount of very high-speed RAM, a larger amount of lower-speed memory, and some technique for loading the fast RAM from the slower one. Sherman and Madsen [SM80] discuss several techniques utilizing dual-ported or double-buffered RAMS to load one part of the high-speed memory while another is being emptied. Such schemes require very wide word access from the low-speed memory to achieve the desired bandwidth or else require pauses in the test sequence to allow for refilling the high-speed RAMS. Albrow [Al83] presents a technique which utilizes one high-speed vector storage RAM

for holding invariant data and another which can be overlaid from the low-speed memory. The idea here is that the invariant memory holds common device-related routines which are used repeatedly in the test sequence, while the variant memory holds the non-repetitive test sequences. In order to transparently overlay the variant memory, the tester must be executing out of the invariant memory for a long enough period of time to refill the variant memory. Such schemes have the disadvantage that they must be manually implemented by the test programmer.

A third class of memory reduction scheme takes advantage of the algorithmic nature of many test sequences. Memory testers are the classic example of such generators. In memory testers, the test patterns must exhaustively search the DUT memory array for sensitivities between adjacent bits or lines of the array. Such algorithms can be efficiently coded in the form of nested loops, but if these loops were to be expressed as in-line vector sequences they would be extremely long. Another common use of algorithmic generation schemes is in serial-scan design structures such as LSSD [EW77]. In such designs, a serial shift register is used to provide read/write access to internal buses of a device, facilitating testing by allowing internal partitioning of the chip. When loading or unloading the serial shift register, only a few of the DUT pins are actively used while the rest remain idle. An algorithmic generator can be employed to convert parallel data from the vector memory to a serial data stream and to provide the necessary clock and control signals. These techniques can result in significant improvement in memory utilization [A183], but require custom hardware for each algorithmic scheme.

Relatively little work has been done in the realm of general purpose compaction schemes for testing purposes. Jackson et al. [JM84] have implemented a scheme which identifies clock-like patterns and repetitive sequences and substitutes a code-word and repetition count for these. Excellent results have been achieved using this technique on certain types of data. However, the drawback of their implementation is that the

compaction is performed on a pin-by-pin basis. By compressing across a group of pins, some of the advantages of an algorithmic pattern generator can be achieved by gaining some degree of vector width/depth tradeoff. Consider for example the LSSD case. Here, some pins are idle and achieve maximum compression while other pins, such as the Serial Data pin, transition at the maximum rate, resulting in poor compression. Assuming that there is no on-the-fly refill capability, the maximum tester run-time is determined by the compression ratio of the worst pin. If a general purpose compression scheme is applied to a group of pins, the effects of active and inactive pins can potentially be averaged, resulting in a longer average maximum run-time.

2.4 Limits of Current Systems

Users of the current generation of integrated circuit testers are becoming increasingly aware of their limitations in testing VLSI devices. As a result of simultaneous increases in both pin-count and device speed, testers are becoming increasingly large and expensive. Size is important not only because of the cost implications, but also because overall performance degrades due to the increased length of the transmission line between the pin electronics and the DUT. When testing ECL or GaAs devices, which have low impedance outputs, it is easy for the tester to maintain a matched impedance environment for both input and output signals to the DUT. Both the tester and the DUT are capable of driving the terminated transmission line which connects them. The only electrical effect of the transmission line is a pure delay which can be calibrated and compensated before actual testing is performed. This is not the case, however, when testing typical CMOS VLSI devices. Because of the limited drive capability of such devices, which is in turn related to their high pin counts, it is typically not possible for the DUT to properly drive a 50 or even a 90 ohm terminated

line. Source termination at the tester outputs can be used to produce high-fidelity waveforms at the DUT inputs, but there is no easy solution to the problem of driving the response signal. The high impedance DUT driver looks into a long open ended transmission line and requires several round-trip delays before it settles to a final value and can be sampled. In current systems, the one-way path length can be as long as 50 cm yielding a round trip delay time of about 5 nanoseconds. This figure is likely to become worse as pin counts increase [Ba83]. One technique for dealing with this problem is to pre-compute the transmission line settling time and factor this into the tester sample delay [Ba84], but to achieve sub-nanosecond timing accuracies, the round trip delay must be reduced. Another consequence of the impedance mismatch between DUT and transmission line is the inability to make accurate measurements under capacitive load. Most devices specify their timing parameters under specific capacitive loading conditions. The lumped capacitive load of the pin electronics at the end of the unterminated transmission line can appear quite large to the DUT output making such measurements difficult or impossible to achieve [Ba84].

The physical size of the test system, particularly the pin electronics, has other influences on the utility of the test system in addition to the electrical problems discussed above. In failure analysis work, for example, it is necessary to connect the test head to the DUT with a good electrical signal environment and still provide microscope and micro-probe access to the circuit. Even with custom fixturing, this is an extremely difficult task. In the near future, device geometries will shrink below one micron making mechanical micro-probing obsolete. When this happens, more exotic techniques, such as electron-beam probing, will become necessary. In this type of system, the test device must be isolated in a vacuum chamber, further complicating the mating of the tester and DUT. The need for smaller testers is evident.

There is clearly a need for testers which are greatly reduced in size from those currently available. The following chapter discusses the basic problems encountered

when taking this approach to tester design. It then introduces the basic concepts for a fully integrated CMOS tester.

Chapter 3

Integrated Testers

3.1 Introduction

The previous chapter discussed some of the features found in large commercial test systems. A high degree of versatility in a tester is valuable since it permits rapid adaptation of the tester to many different types of test circumstances. The problem is that, in the current generation of machines, versatility implies a large and expensive system. Because of their size and cost, these machines are typically used only in high volume production testing, leaving the average designer with poor facilities for design debug and diagnostic testing. This chapter develops the basic concepts for building integrated test systems that are small and low cost, but which still preserve many of the useful features that are found in the larger test systems. While this may seem to be contrary to established principles, it is one of the aims of this thesis to prove that there is a non-linear relationship between tester size and complexity: when the system is forced to occupy a much smaller physical space, many of the engineering problems that the designers of larger systems must face simply disappear, allowing a much cleaner system design.

3.2 Don't Think Big

A survey of the market today for high-end test systems, would find many mainframe-sized testers, each costing on the order of a few million dollars. The basic structure of these machines is essentially the same: several racks full of electronics to generate the test vectors, connected via a large number of high speed cables to a bulky pin electronics head, which in turn interfaces to a single small test device. The obvious question is, "Is all of this really necessary?" Indeed, is it the case that designers of such equipment have built themselves into a self-fulfilling prophecy? In setting out to create a world-class tester, they have assumed, based on previous designs, that it would be a physically large machine. The large size imposes a number of problems having to do with signal skew, loading, accuracy, and distribution that require additional hardware and software subsystems to correct. As the number of subsystems increases, the communication and control costs increase as well, leading to further system complexity. In the end, the tester must be a physically large machine to accommodate all of the overhead hardware, thus justifying the designers' initial assumption. In order to test this hypothesis, the initial assumption must be changed. What would a VLSI test system be like that was designed to be as *small* as possible?

3.3 Think Small

In order to build the smallest possible system, the obvious choice is to employ high density VLSI technologies. The number of transistors that can be implemented in a bipolar technology is fairly limited due to power dissipation, so a higher density technology like CMOS seems to be the likely candidate. CMOS has other inherent advantages that make it a good technology choice as well: it supports low power,

is relatively easy to fabricate, and is the dominant technology in use for ASICs today. Also, the tools required for chip development are widely available and there is a substantial body of design experience available. Thus, there is some hope that the design can be 'portable,' so that as improved fabrication processes become available, the performance of the tester can readily track the technology. The main disadvantage of choosing CMOS is that it is not a particularly fast technology. If the principal application of the tester is for testing CMOS ASIC components, then the technology of the tester is only on par with that of the DUT. This can be turned into an advantage, though, for if the desired performance is achievable, then the tester technology relies only on the same technology as the device to be tested. No other more exotic technology is required, thus allowing the tester technology to track that of the DUT. Let us put aside the speed difficulty for now by assuming that it can be handled through clever circuit design and parallelism. It will be revisited in Chapter 4.

High-speed interconnect in any system is a difficult problem. In commercial test systems, one typically finds large masses of bulky coaxial cables tying various pieces of the system together. This type of interconnect cannot be tolerated in a test system that is to be as small as possible. The best way to avoid it is to constrain all of the high-speed components of the system to fit together on a single printed circuit card where signal impedances can be carefully controlled and properly terminated. Constraining the system to a single card has other advantages, as well. A crucial parameter of any test system is the physical length of the transmission line between the pin of the DUT and the driver/receiver electronics of the tester. The length of this path should allow several round-trip propagations within the minimum edge risetime, in order to achieve good signal fidelity for both driven and received waveforms. Allowing several round-trip propagations provides sufficient time for the signal to settle so that signal termination is not needed. The short path length also minimizes the

capacitive loading on the DUT output. Another advantage of placing all of the high-speed portion of the system on a single board is that it drives the system towards a clean, low-speed interface to the outside world. Presumably, the system is controlled by a host mainframe, workstation, or PC, so eliminating all real-time constraints from the host's interface ensures that this part of the system can be simple and low cost as well.

With the single board system concept in mind, attention can now be focused on the individual components that make up the building blocks of the system. There are three main functions that these components must implement: timing generation, pin drive/sense, and vector storage. In most test systems, timing generation is considered to be a relatively expensive part of the tester design. Typically, a test system will have a limited number of timing generators (8 to 32) and distribute these via a switching matrix to the various pins of the system. This is a classic case where thinking small and applying new design techniques can improve the overall system. The switching matrix in the typical system represents considerable overhead since it must perform the function of an $m \times n$ crossbar switch, where m is the number of timing generators and n is an integer multiple of the number of pins in the system (each pin uses several timing edges). When coupled with the fact that the clock distribution must be performed with near zero skew, it is clear why the switching matrix is complex. To eliminate it from the design, a separate timing generator is necessary in each place where one is needed. In conventional systems, the timing generators are highly accurate, providing timing resolution on the order of a hundred picoseconds or so. This necessitates a considerable amount of logic working at frequencies in the range of hundreds of MHz, making it impractical for high density integration and in turn difficult to replicate for each pin. An alternative solution to this problem is found by removing the restriction that the timing generators be very accurate. Rather, provide only one accurate timing source in the system and distribute it uniformly to

all elements. Then, through calibration and feedback, adjust each of the local timing generators to have precise delays. This approach completely eliminates the switching matrix and all but one of the high speed, high accuracy, timing generators. As shown in the following chapter, the local timing generators can then be highly integrated using conventional CMOS technology.

The second functional element of the design is the DUT pin drive/sense electronics, or pin electronics. The pin electronics portion of a tester is the area of tester design that can be most easily mapped directly into CMOS circuitry. The pin electronics must be capable of driving an output waveform to the DUT with known timing and voltage level characteristics. It also must be able to capture a response waveform from the DUT and compare it with an expected value. Thus the key elements of the pin electronics are the output driver and input comparator. It is essential that these circuits be located as close as possible to the DUT for rapid settling. For an unterminated signal, the round trip delay on the wire connecting the pin electronics and DUT should be less than the minimum rise time of the driven or sensed signal. Given a 2 ns edge rate, this limits the distance to about 10 cm. Some tester systems provide additional features in the pin electronics such as programmable load devices for voltage and current. In large testers, these devices allow rapid prototyping of external load conditions, which is important when tester time is expensive. In a small system, perhaps dedicated to a particular device, it is feasible to implement loading with external individual components, thereby reducing the amount of integrated analog circuitry in the pin electronics.

The last important element of the design is the vector storage unit. The vector storage unit is a special purpose memory that holds the test vectors that are to be applied to the DUT. In the worst case, the memory is several times as wide as the maximum number of pins to be tested, since several bits are needed to represent the vector information needed by each pin in every cycle. The trend in commercial

testers in recent years has been to greatly increase the depth of the vector storage, thus allowing much longer test sequences. There are two reasons for this trend: the increasing number of transistors per chip, which require more vectors to test, and the increasing use of automatic test pattern generation programs, which are not as good as hand-coded vectors in terms of coverage efficiency. In an expensive system where tester time is costly, it is important to achieve maximum throughput by holding all of the test vectors in the vector storage unit at once to avoid the overhead of reloading. This is not the case, though, for a low cost system. As long as sufficient vector memory is available to prevent excessive reloading and the reloading time is relatively small, the cost of the tester time is negligible. In the case of dynamic test devices, some additional cost is incurred since each buffer load must be capable of restoring the DUT state at the beginning of the load. The alternative is to provide the tester with the ability to run a "keep alive" loop while the vector storage is being reloaded, although this may not always be a feasible alternative for the DUT. In either case, the overhead should be small.

To examine the feasibility of this approach, consider an example test system with 256 active DUT pins, vectors of four bits per pin, one million test vectors, and a vector buffer 8K deep. A large tester, to be sure. Assume that vector bits can be loaded 16 bits at a time and that each transfer into the vector memory takes 10 microseconds (software loop). Each buffer then consists of $256 \times 4 \times 8K = 8$ Mbits, and the transfer rate into the buffer is $16 / 10^{-5} = 1.6$ Mbits/second. So a single buffer reload takes only 5 seconds. To perform 16 buffer reloads ($1M/8K$), requires 80 seconds. Thus, even in the worst case situation, the time to run a million vector test is still only a little more than a minute. With a minimal amount of external DMA support, the transfer time could be reduced by nearly an order of magnitude, bringing the test time down to a few seconds.

Reducing the amount of memory required by the vector storage unit to the absolute minimum is an important issue because the size of the RAM determines the feasibility of including the vector storage on the same chip as the rest of the system logic. The alternative is to use external commercial memory components. Including the storage on the chip avoids a number of complexities encountered with board level RAMS, the primary problems being bandwidth and delay. Again, consider the case of a 256 pin tester. Assuming that each pin of the tester will require one to four RAM bits, depending on the encoding scheme used, the tester requires a RAM data word width in the range of 256 to 1024 bits. Each bit of the data word requires a pin on a tester chip, a pin on a RAM, and some length of printed circuit trace. This greatly increases the pin count and package size of the tester chip, the spacing between tester chips due to the large number of etch traces on the board, and the number of external components. The widest RAMS currently available are 8-bit wide parts, so between 32 and 128 external components are necessary. This is already quite a severe limitation for a single board tester. The situation clearly worsens when higher pin count testers are considered. The limitations on word width also have a direct impact on RAM bandwidth. Due to component count, multi-banking the RAM to improve data bandwidth is extremely expensive, so a single RAM bank must be capable of delivering its data at the maximum tester cycle frequency. On-chip RAM, on the other hand, avoids all of these problems. Multi-banked RAM is not a problem since bit multiplexors can be placed directly on RAM outputs. With proper floorplanning, signal routing can be virtually eliminated, and extra pin count and component count are eliminated.

The ideal device for building a single board tester would incorporate all three of the functions of vector storage, timing generation, and pin electronics for several tester channels on a single chip. In addition, it would contain the necessary interface and control logic so that it would interface directly, via a low speed interface, to the host computer and directly, through short traces, to the DUT. A number of these devices

could be used in parallel to build testers of arbitrary width. A small number of clock and calibration signals would be common to all elements of the system, but since they are relatively few in number, a good deal of care could be taken in their generation and distribution. The master timing source would either be a custom device or a small amount of board level logic since only one copy would be required. The system would easily fit on a probe card for wafer analysis and in-circuit debugging. It could also be added as a single board to a workstation for packaged-part testing. Such a tester could have good performance characteristics, yet still be small and low cost.

3.4 Limits

Before delving into the detailed design of the system, it is worth considering what the fundamental limits of the system may be. The two most important parameters of any tester design are “how fast” and “how accurate.” In its basic form, a tester architecture can be viewed as a simple serial machine. At one end, test vectors are fetched from a vector memory, perhaps processed in some manner to add timing or control information, and then presented to the DUT. The response, meanwhile, is captured from the DUT output pins and compared to a similar reference data stream. Every vector operates on the DUT in isolation and there is no communication between non-sequential pipeline stages in the data path. Since there are no latency requirements on the delivery of data from vector memory to DUT, or for sensing when an error occurred, the ultimate data-handling speed in such a machine is limited only by the extent to which the processing element delays are pipelined. In the case where the memory data path is of limited width, as with off-chip RAM, the memory fetch time is a limiting factor, as well. Thus, given enough die real estate, speeds close to the maximum register transfer frequency of the technology could be achieved. In a practical system, however, the restriction limiting inter-pipeline stage communication

is difficult to maintain, and excessive pipelining can be impractical. In Chapter 5, the idea of data compression is introduced for improving the efficiency of on-chip RAM storage. This breaks the one vector per RAM word assumption and forces feedback to determine the starting position of the next word of data. Chapter 5 also introduces the requirements of branching, which places additional demands on the vector delivery system. From the test programmer's point of view, the ideal branch operation has a zero cycle latency, requiring that the succeeding word in the Force Data pipeline be modifiable by the current DUT response. This is difficult to do, since it requires either the elimination of all pipelining from the system or else a complex branch prediction mechanism. Some compromise must therefore be made between branch latency and tester performance.

To get some feel for the actual speed that may be attained from the tester for a given technology, the best method may be to compare it to some other device which has been already implemented in that technology. A good choice is a microprocessor CPU. A microprocessor cycles through the operations of fetching instructions from memory (e.g., an on-chip cache), decoding them in a pipelined fashion, and producing some result. These are the same functions performed by the vector generator in a tester. A great deal of effort typically goes into a new microprocessor design, which would tend to positively bias the maximum speed, but perhaps this is balanced by the relative simplicity of the tester data path. For a 2μ CMOS process, it is reasonable to expect a microprocessor to operate in the 25 MHz range. This speed should scale fairly linearly as device geometry shrinks, as long as transistor velocity saturation effects can be avoided. Similar speeds should be attainable in a tester.

A performance aspect that is equally important to cycle speed is edge placement accuracy, both for output waveform transitions and input sampling. With a feedback calibration scheme, near perfect accuracy can be assumed in the reference delay generator since only one is required in the system. The edge placement limitation

then depends primarily on the capture uncertainty of the phase comparator and the phase noise and accuracy of the local delay generator. Through repetitive sampling techniques [Ke86], the noise introduced by the phase comparator can be arbitrarily reduced, leaving the local delay generator as the principal source of noise. The phase noise added by the delay generator is difficult to characterize, as it is heavily dependent on both the circuit implementation and the system packaging. The basic gate delay in CMOS technology is strongly influenced by supply voltage, so supply noise and printed circuit board layout are important issues. Temperature variations also introduce a low frequency noise component, so some form of regulation or feedback is needed to insure long term calibrated accuracy.

Though there are many technical issues left to be resolved, no fundamental reasons exist that would prohibit the building of a tester with performance characteristics sufficient to test nearly any device fabricated in the same technology.

3.5 Summary

Current generation integrated circuit testers are large and expensive. By taking the radical new approach of integrating most of the features of large testers onto a single chip, a much better cost/performance tradeoff can be achieved. The following chapters describe the detailed design of a single chip tester system based on the principles outlined here.

Chapter 4

Pin Electronics

4.1 Introduction

In the previous chapter, the structure of integrated test systems and the inherent advantages of the integrated approach were presented. In this chapter, we begin to investigate some of the issues encountered in the design of an integrated tester chip. Part of the problem of building a fully integrated VLSI tester is to integrate the functions performed by the pin electronics. The design of integrated pin electronics circuitry cannot be found by recasting existing designs into a new technology. Rather, the fundamentally different nature of the target technology requires a wholly new approach to both the architecture and circuit design. This chapter presents an architecture for building CMOS pin electronics circuitry. An experimental four channel pin electronics device was designed using these ideas to prove the validity of the concepts. The experimental device performed according to design specification with only a few minor problems.

A block diagram of a single channel of the pin electronics architecture is shown in Figure 4.1. It consists of three main sections: Force, Acquire, and Calibration. The Force section takes the stimulus information of the test vector, combines it with timing

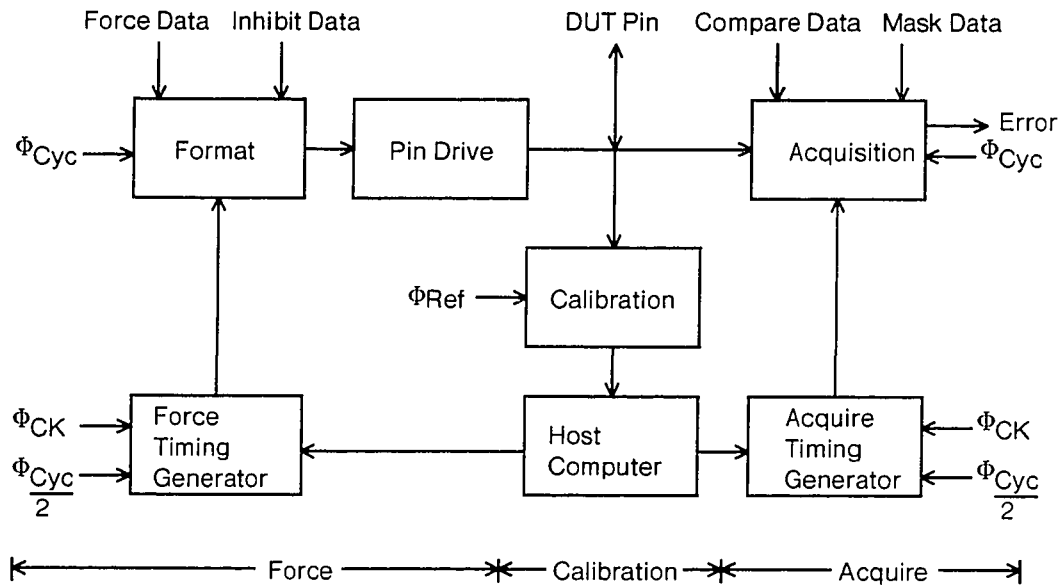


Figure 4.1: Pin electronics block diagram

information, and drives the input pins of the DUT. The Acquire section samples the output pins of the DUT at a specific time and compares the results with the information supplied by the Acquire vector to determine if an error has occurred. The Calibration section closes the loop on both the Force and Acquire units to provide high accuracy timing. Each timing generator in a channel is independent of other channels on the chip. Each channel can also independently select high or low drive voltage and input threshold voltage, providing a true per-pin architecture.

4.2 Force Section

The Force section consists of three sub-blocks: the Force Timing Generator, the Format unit and the Pin Drive circuitry. The Force Timing Generator provides a timing control pulse which determines where the edge transitions will occur in the

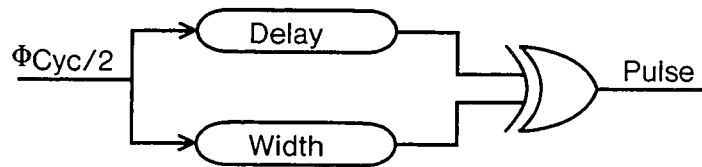


Figure 4.2: Force timing generator

DUT waveform. This pulse determines the transition times of the waveform created by the Formatter. The Pin Driver buffers the low-level internal signal up to the external drive level and also adds the analog high and low level voltage information. The setup and calibration of the timing pulse, format mode, and drive level are all done statically before the test is run, although there can be a number of register sets for each of these units that are initialized with different calibration and format parameters. Dynamic switching would then be possible between various timing and format modes during test execution.

The Force Timing Generator consists of two independent delay generators: one for pulse delay and one for pulse width (Figure 4.2). Of all the functional elements in the pin electronics architecture, the element that has the greatest impact on the die area is the delay generator. Three of these generators are in each channel (one additional unit in the Acquire Timing Generator). Each generator must be capable of covering a wide delay range, from 0 to 1 μ s, with a resolution of less than 1 ns. Because of this large “storage” requirement, the generators occupy a considerable fraction of the die area, so an efficient scheme is needed for implementing the function. There are two basic approaches to the problem: either build high accuracy delay generators using techniques such as PLL delay lines [JB87], or use circuits that have low absolute accuracy but are stable and can be easily calibrated. The latter approach is more suited to the CMOS implementation because the projected die size is better, and also

because of the ability to calibrate the intrinsic delays caused by other elements of the system, such as the Format Generator and Pin Driver electronics.

For full flexibility in shaping the output waveform, the timing control pulse must cover the maximum possible range of duty cycle and edge placement within cycle boundaries. Allowing the user to place edges near the beginning or end of the tester cycle makes the design of the pulse generator difficult because of the intrinsic delays of the logic. In addition, producing pulses with very small or very large duty cycles is difficult since delay lines tend to distort narrow pulses. Both of these problems become more serious when a relatively low speed technology like CMOS is used. The diagram of the Force Timing Generator in Figure 4.2 shows the circuit that was used to overcome these problems. Two programmable delay lines are used, one for pulse delay and one for pulse width. The delay lines consist of three different delay elements: a shift register for coarse delay adjustment, an inverter chain for finer resolution, and set of ratioed inverter delays for very fine (sub-nanosecond) resolution. The input to both delay lines is the square wave clock, $\Phi_{CYC/2}$, whose period is one-half that of the tester cycle. To form the timing pulse, one delay line is set to the desired delay time and the other to the desired delay plus the desired width. The outputs of the two delay lines are then exclusive-ORed to produce a double frequency pulse train of the required shape. This technique eliminates the problem of narrow pulses since the delay lines always see a 50% duty cycle signal. The periodic nature of the pulse traveling through the delay lines provides a method of placing transitions near the tester clock edges. For example, if an output transition is desired near the beginning of the clock cycle at a point less than the inherent delay, the timing pulse is delayed into the next tester cycle. Since the pulse train is periodic, the result is the same as building a pulse generator with zero inherent delay. Thus, the circuit makes it very easy to position a pulse anywhere in the cycle: there are no dead bands.

The delay elements and the XOR operation are not perfect and do introduce some

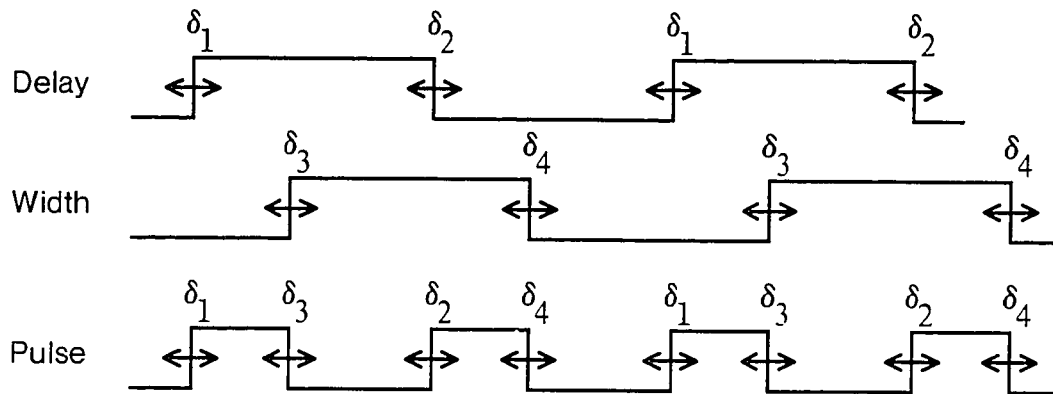


Figure 4.3: Edge deviation

edge timing distortion in the pulse waveform. The effect of XOR-ing two slightly distorted 50% duty cycle waveforms is to produce a waveform whose basic period consists of a pair of pulses with each edge of the pair deviating slightly from the desired timing (Figure 4.3). The distortion can be corrected during calibration by use of the high resolution edge adjust units at the final stage of the delay elements. As seen in Figure 4.4 and Figure 4.5, the edge adjust units consist of a simple NAND gate that applies the waveform to be adjusted applied directly to one input and through a high resolution delay element to the other input. The effect of the gate is to allow falling input edges to propagate with minimum delay and rising edges to propagate with a controlled delay. The output of the NAND inverts the original signal so that concatenating two such units gives individual control over both rising and falling edges of the waveform. Since both the Delay and Width generators have independent controls on the rising and falling edges of each of their respective output waveforms, the XOR output edges, δ_1 through δ_4 , can be independently calibrated as well.

Each of the three different delay types employ a different technique to achieve the desired delay and resolution required by that stage. In order to obtain relatively long but stable delays, a shift register is used with an external clock to supply stable reference (Figure 4.6). The final output stage of the shift register allows selection

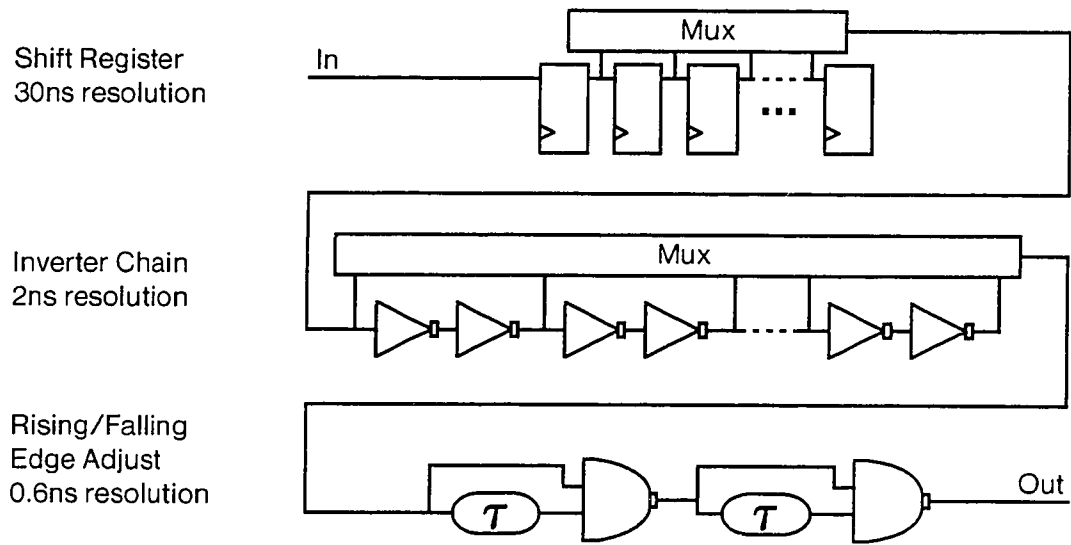


Figure 4.4: Timing chain

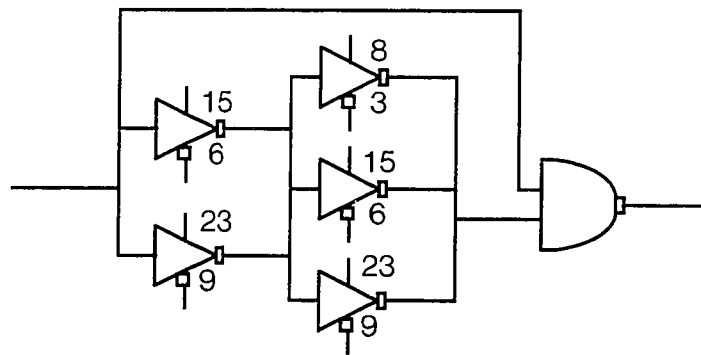


Figure 4.5: Edge adjust (transistor widths in microns)

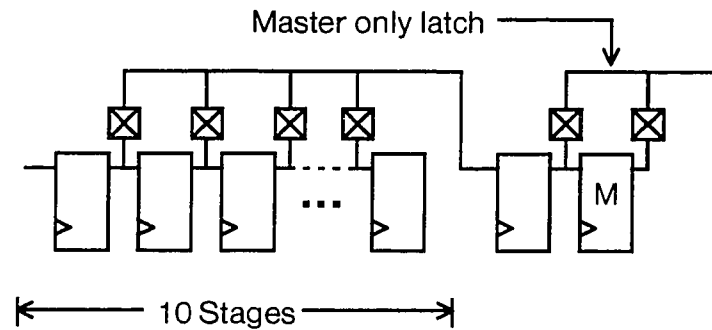


Figure 4.6: Shift register

of either a normal rising edge triggered D-flop output, or a falling edge triggered master-only stage. This provides an overall resolution of one half of the clock period. The inverter chain stage uses the minimum delay through a pair of inverters as a compact means of obtaining moderate resolution (approximately 2 ns) over a range that covers slightly more than one half the shift register clock period (40 ns). Two additional stages in the inverter chain allow the selection of a constant 1 or constant 0 at the output. This is useful during calibration to make each of the internal delay elements independently observable. The final delay type is the differential inverter path. Here a resolution is achieved which is better than the minimum gate delay of the technology by utilizing the difference in path delay through pairs of carefully sized inverters (Figure 4.5). The incremental delay obtainable using this method is limited by the phase noise of an inverter, which is heavily dependent on V_{DD} noise. The gate widths shown in the schematic were derived using Spice simulations and achieve a resolution of about 600 ps (Figure 4.7).

A delay element using the techniques described here was incorporated into an experimental pin electronics device built to verify the validity of these design principles. Upon testing this circuit, the range of the edge adjust units was found to be insufficient to correct the skew errors (δ_1 through δ_4) introduced by the two delay

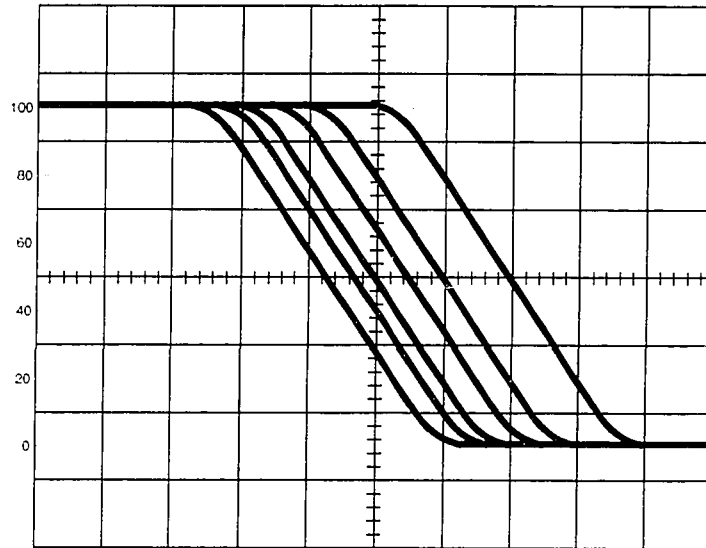


Figure 4.7: Replica of an oscilloscope photo showing the edge resolution waveform (1 ns/div.)

generators. To remedy this situation, the delay range of each of the adjusters was extended from 2.5 ns to 8.5 ns by adding a four stage inverter chain as shown in Figure 4.8.

To accurately probe the frequency limits of the DUT, the pin electronics needs to provide continuous operation over a wide range of frequencies. This presents a

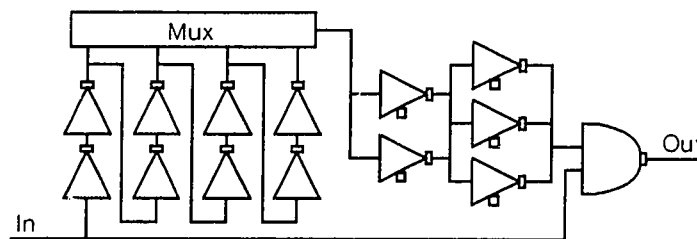


Figure 4.8: Edge adjust unit with improved range

problem in the multi-stage delay generator chain. For the scheme to work, the *range* of delay provided by each of the delay types must be greater than the *resolution* of the previous stage. A problem occurs in the shift register delay stage as the tester frequency is reduced from the maximum. The resolution of the shift register decreases with the clock frequency, but the range provided by the following stage, the inverter chain, remains constant, since it is not tied to the clock frequency. To solve this problem, the shift register clock is selected to be a multiple of 1, 2, 4, or 8 times the cycle clock frequency depending on the cycle frequency. By doing this, the resolution of the shift register is always constrained to be within the range of the inverter chain. Table 4.1 shows the relationships between the tester cycle clock, shift register clock (SR), inverter chain (IC) and differential inverter path (DIP) elements.

| Cycle Freq. (MHz) | Φ_{CK}/Φ_{CYC} | SR Res. (ns) | SR Range (cycles) | IC Res. (ns) | IC Range (ns) | DIP Res. (ns) | DIP Range (ns) |
|-------------------|------------------------|--------------|-------------------|--------------|---------------|---------------|----------------|
| 30-15 | 1 | 15-30 | 9.00 | 2 | 40 | 0.5 | 8.5 |
| 15-7.5 | 2 | 15-30 | 4.50 | 2 | 40 | 0.5 | 8.5 |
| 7.5-3.75 | 4 | 15-30 | 2.25 | 2 | 40 | 0.5 | 8.5 |
| 3.75-1.88 | 8 | 15-30 | 1.13 | 2 | 40 | 0.5 | 8.5 |

Table 4.1: Delay element ranges and resolutions

As can be seen from the table, below 1.88 MHz, the on-chip delay elements no longer have sufficient range to cover a full tester cycle. A broader band coverage can be achieved by adding a fourth delay type to the front of the delay chain consisting of a shift register clocked at one-eighth of the successive shift register frequency to provide coverage well into the kHz range.

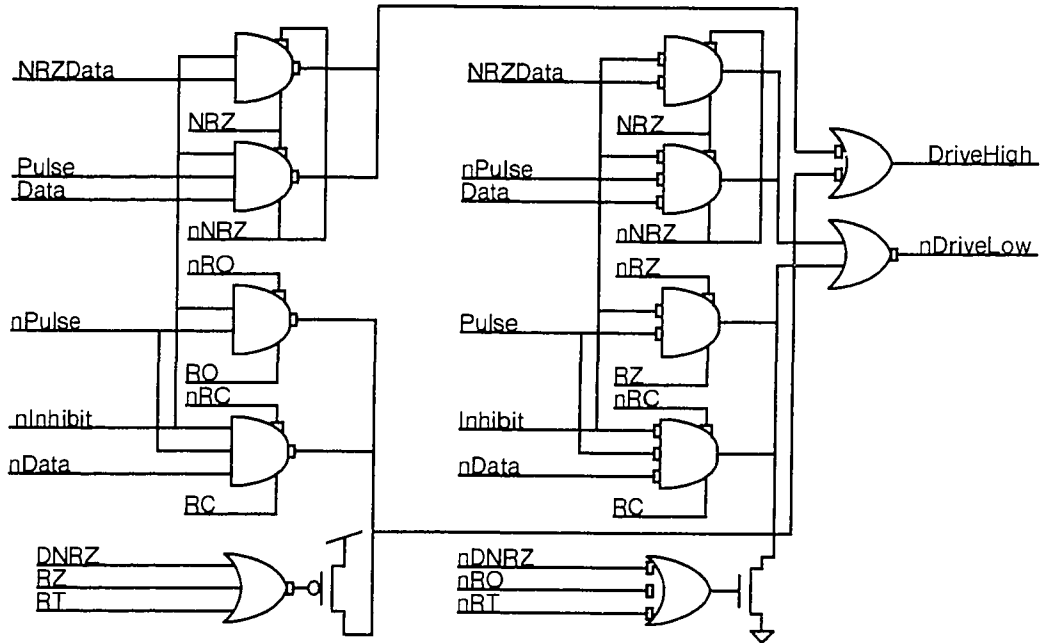


Figure 4.9: Schematic of format generator

4.3 Formatter

The Formatter combines the stimulus data of the Force Vector with the edge placement information of the Timing Generator. This data/timing mixing is performed by the logic shown in Figure 4.9 in one of five selectable modes: Non-Return to Zero (NRZ), Return to One (RO), Return to Zero (RZ), Return to Tri-State (RT), and Return to Complement (RC). The formatter is designed such that all paths through the generator have equal delay, permitting the format mode to be changed during test vector execution without affecting the timing calibration. Figure 4.10 shows the oscilloscope waveforms produced by the experimental pin electronics chip generating

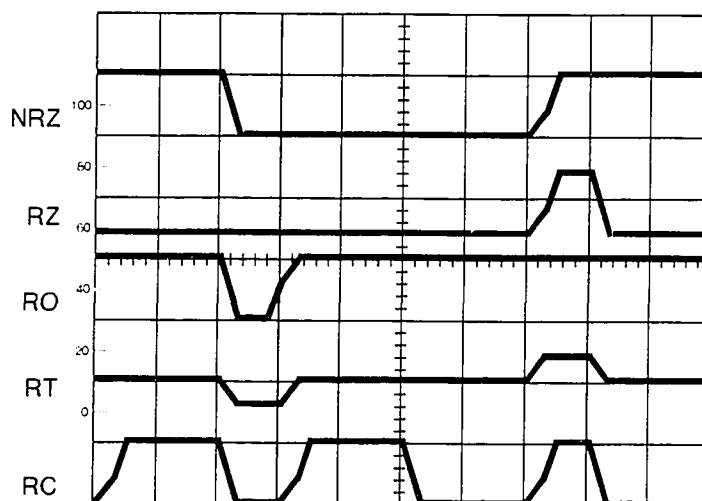


Figure 4.10: Replica of an oscilloscope photo of the PE chip generating five output formats at 20 MHz (10 ns/div.)

each of the five output formats¹. As seen in this figure, the NRZ format is a non-pulsed mode; that is, a transition is made to the level specified by the input vector after a given delay and that level is maintained until the next tester cycle. There is at most one data transition per cycle. This mode is useful for simulating edge-triggered events such as register or counter outputs. All of the other four formats are pulsed modes: they start at a default level (ZERO, ONE, or tri-state for RZ, RO, and RT, respectively), transition to the desired level after a given delay, maintain that level for a set width, then return to the starting level. In operation, these modes allow the tester to simulate events at rates higher than the basic tester cycle rate would allow. For example, consider the problem of generating the two-phase non-overlapped

¹The kink in the rising edge of the output waveform was due to an unexpectedly long wire (generated by an automatic router) between the pre-drive and final-drive stages of the output buffer. The extra capacitance introduced enough skew to cause a few nanoseconds of overlap between the pad Drive High and Drive Low signals. This problem was corrected in a later version of the device.

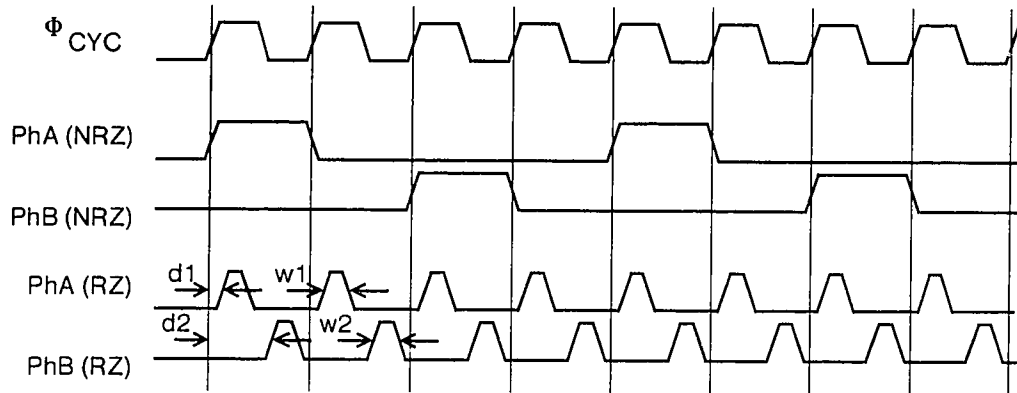


Figure 4.11: Non-overlapped clock example

clock used in many MOS systems. The simple method would be to use successive tester cycles to generate the four clock states (Low-Low, Low-High, Low-Low, High-Low). A better method would be to use two channels in RZ format with delays and widths adjusted appropriately to form the desired non-overlapping pulses within a single tester clock cycle (Figure 4.11). An additional advantage of this method is that the clock duration and non-overlap times are individually programmable with high resolution. The RO format could also be used in such an example to generate the complementary clock.

The RT mode doubles the channel's IO bandwidth in much the same manner that used by the RZ and RO modes to increase the output event rate. Consider again the non-overlapped clock example. Suppose the DUT latches its inputs on Φ_A and supplies a response on Φ_B . RT mode can be used to drive the force data during Φ_A and then inhibit the output during Φ_B when the DUT is driving the bus. The sample clock in the channel would be programmed to capture the DUT response at the appropriate point during Φ_B . This situation is the only case where both of the test vector bits Force Data and Compare Data are needed in the same tester cycle.

The primary use of the RC mode is for making set-up and hold measurements. In RC mode, the default level is the complement of the data to be driven. After

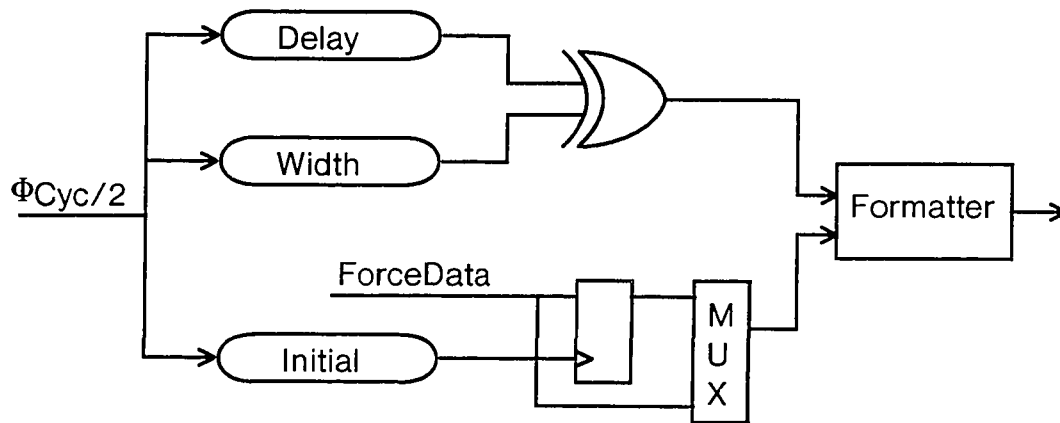


Figure 4.12: Addition of a fourth delay generator for calibrating RC mode

a programmed delay the output transitions to the value specified by Force Data maintains that value for the specified width and then returns to the complementary value. By adjusting the delay and width of the drive pulse to exactly meet the specified set-up and hold requirements of the DUT, the device will fail if it exceeds specifications. The RC mode differs from each of the other four modes in that it can cause up to three data transitions per cycle (RT can have three or even more transitions per cycle, but only two are caused by the pin electronics driver). One transition occurs at the start of each cycle to assume the complementary drive value, and two more occur when the output pulses to the drive value and back to the complementary value. Since the timing generator can only adjust the delay and width, the initial edge is neither adjustable nor calibrated. One solution to this problem is to add a fourth delay generator to the pin electronics circuitry as shown in Figure 4.12. The only use of this generator would be to position the initial transition in RC mode. The die area required for this fourth generator is fairly large, particularly in the 2μ technology used for the experimental pin electronics chip and the demonstration single-chip tester discussed in Chapter 6. Also, when the RC mode is used as intended

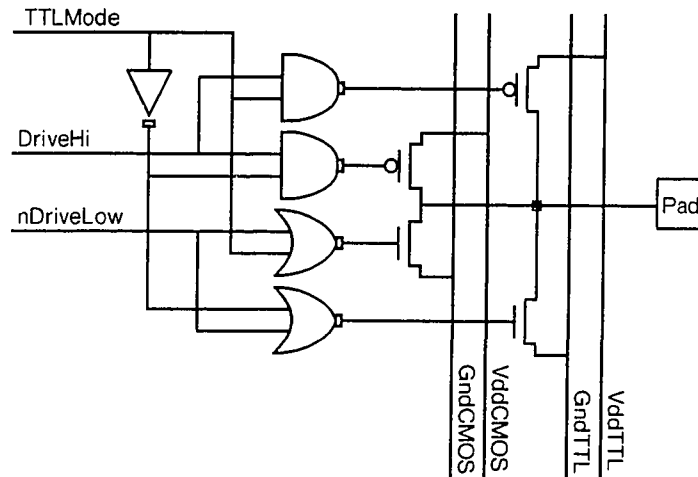


Figure 4.13: Pin driver circuitry

for making set-up and hold measurements, this edge would fall completely outside of the capture window of the DUT, so the exact placement of the transition is not crucial. For these reasons, the fourth delay generator was not included in either design. A compromise solution is possible, which requires very little die area. By simply clocking the flip-flop in Figure 4.12 with the inverse of the tester cycle clock, the initial data transition can optionally be moved from the beginning of the cycle to mid-cycle. This allows the deadband to be moved around, so RC measurements can be near the cycle boundary.

The stimulus waveform generated by the formatter must be amplified to drive the potentially high capacitance load of the DUT. The pin driver circuitry provides buffering to drive loads up to 50 pF. In addition, it provides a selection of two high-level and two low-level drive voltages (Figure 4.13). The two levels allow some DUT pins to be driven at CMOS levels, while others are being tested for TTL compatibility. The dual level pads essentially consist of two tri-state drivers connected in parallel.

The output levels are set by choosing which driver to enable.

4.4 Acquire Section

The Acquire portion of the pin electronics circuitry is responsible for capturing data from the DUT and determining if it matches the desired response. Both the data value and output voltage level must be checked. In the Force Timing generator, both delay and width timing control are needed, but in the Acquire Timing generator only control of the sample delay timing is necessary. This allows the Acquire Timing generator to be simplified to a single delay line. The input to this delay line is still $\Phi_{CYC/2}$, as it was for the Force Timing generator. This introduces a problem: the frequency doubling effect of the XOR gate is not realized, so the delay line output is only half the desired cycle frequency. Since the die area consumed by the sample and comparison circuitry is much smaller than that of a second delay line, the solution is simply to build two sampling circuits: one which operates on the positive transition of the sample timing clock and one which operates on the negative transition. A beneficial side effect of this technique is that the data at the output of the two comparators is available for a full tester cycle. This greatly simplifies the problem of resynchronizing the data between the sample logic and the data comparison pipeline. Since the phase of the captured signal is known, an optional pipeline stage clocked at the midpoint of the cycle ensures that the data can be captured at a stable point in the cycle (Figure 4.14). In the data pipeline, the acquired value is compared with the Expect Data from the test vector. If the comparison fails and the Mask bit from the test vector is not asserted, the external Error bit is asserted for the cycle.

The front end of the acquisition circuitry consists of a clocked analog voltage comparator. The positive edge triggered version of this circuit is shown in Figure 4.15. The circuit is very similar to a dynamic RAM sense amplifier. The heart of the circuit

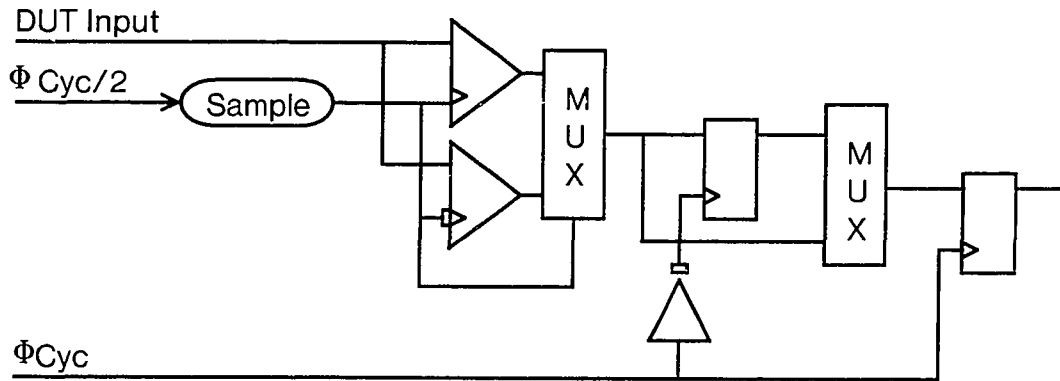


Figure 4.14: Acquisition and synchronization logic

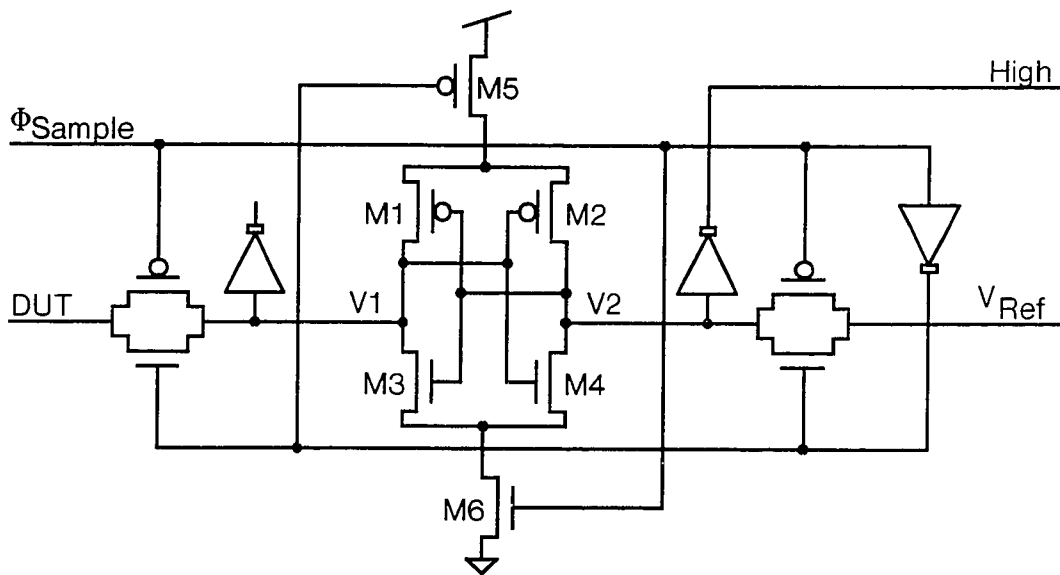


Figure 4.15: Clocked analog voltage comparator

is a cross-coupled inverter pair consisting of transistors M1 through M4 which can be decoupled from the power supply rails by transistors M5 and M6. When the sample clock is low, the inverter pair is disconnected from its supply rails and the two transmission gates couple the voltage on the DUT test pin and the externally supplied reference voltage V_{REF} to the sensing nodes, V1 and V2. After a delay determined by the Sample Timing Generator, the sample clock rises, isolating the sensing nodes and connecting the supply rails to the cross-coupled pair. Depending on the initial conditions impressed on the sensing nodes while the clock was low, the cross-coupled pair will quickly fall into one of two stable states (1, 0 or 0, 1). This value remains latched in the sense amplifier until the next time the sample clock falls. The comparator accuracy using this circuit is better than 50mV.

4.5 Calibration

Calibration is a key feature of the pin electronics circuitry. The on-chip timing generators are very poor in terms of absolute or even relative accuracy, but they are highly stable for a given delay setting. In order to achieve sub-nanosecond accuracy, each timing generator must be calibrated to an external precision timing reference. For the experimental system, a GPIB programmable pulse generator and a small amount of board level logic were used to synthesize the three synchronous clocks. The reference clock was derived from these clocks using a commercial 8-bit programmable delay-line device. This system provided 1 ns step resolution and fair (5%) overall accuracy.

An important aspect of the calibration system is the amount of on-chip circuitry required. Since this circuitry must be present in each channel, it is necessary to keep the real estate required to a minimum. To perform the calibration function, the phase of a transition on the DUT pin needs to be measured with respect to a transition on the reference clock. All that is needed is a digital Early/Late indication of the

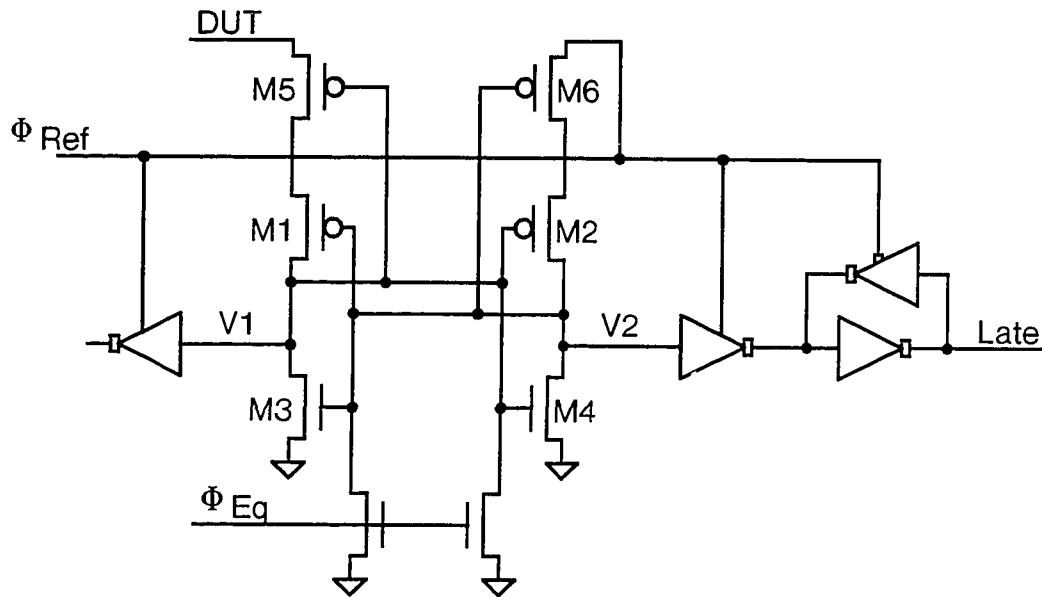


Figure 4.16: Phase detector

comparison result. By searching for the Early/Late transition using progressively finer resolution delay steps, the edge of the reference clock can be found to within the minimum step-delay resolution of the on-chip delay generator. The only per-channel circuitry needed to perform this operation is a digital phase detector and some logic that allows the host computer to read the comparison result. Figure 4.16 shows the schematic of the phase detector that was used on the experimental pin electronics chip. The schematic shown is rising-edge sensitive; an additional circuit of complementary design is used to calibrate falling edges.

Like the clocked input comparator, the detector uses a modified differential sense amplifier design, M1 through M4, to detect small timing differences between the DUT and reference waveforms. Instead of precharging the sense nodes V1 and V2 to some initial offset, the nodes are both precharged to ground. The power supply for each of the cross-coupled inverters is connected to an input signal via M5 and M6. The input which rises first pulls up on its output node which causes the other output node to

be clamped. The additional two p-channel transistors at the top of the cross-coupled stack act as diodes that allow the output of the detector to maintain the comparison result even when one of the input pulses is very narrow. The output of the comparator is latched when the reference clock falls so that the result can be sampled by the host computer.

This circuit exhibits very good phase comparison accuracy, but suffers from one fatal flaw, which was discovered during testing of the experimental device. When the DUT output is driving a highly capacitive load or a long signal trace on a PWB, a fair amount of overshoot can be seen on the DUT pad due to signal reflection. This overshoot appears as a greater V_{DS} bias on the p-channel devices of the cross-coupled sense amplifier and results in a phase offset of several nanoseconds in the comparator. Termination of the signal line could be used to reduce the signal reflection effects, however, this is not a very desirable solution. A better approach is to use a simple edge-triggered D-type flip-flop for phase comparison. The implementation shown in Figure 4.17 exhibits superior rejection of overshoot effects and requires only a slight increase in layout area. By balancing the gate loadings, the circuit has a uniform offset of 350 ps between the clock and data inputs when sampling both rising and falling edges. Since all edges in the system are calibrated using this circuit, the offset does not appear in the output waveform, and minimal channel to channel skew is achieved.

For optimal system performance, close attention to skew control is mandatory. The DUT input to the calibration circuit is sensed directly at the DUT output pad so that all delays and skews associated with format generation and pad drive circuitry are compensated. In order to minimize inter-channel skew, a metallic connection is maintained between the reference clock input pad and all channels on the chip to eliminate propagation delays. In a multi-chip system, careful attention to board layout is necessary to assure uniform propagation delays between signal source and

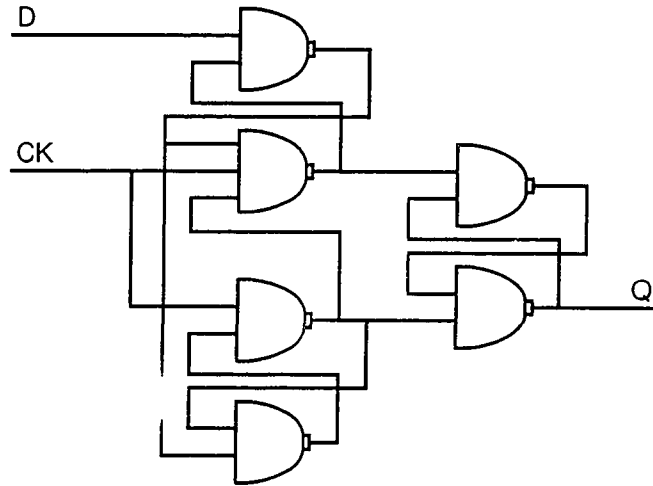


Figure 4.17: Schematic of D-type flip-flop

destination. With these techniques, the output waveform can be calibrated to within the resolution of the timing generators, which is about 600 ps.

The calibration procedure consists of setting the global timing source to the transition time of the channel-edge to be calibrated. Then, a short sequence of test vectors are run through the output channel to allow the phase detectors to determine the relative timing of the output transition with respect to the reference. The calibration loop is then closed by allowing the host computer system to read the state of the phase detectors. This information is then used to compute a new setting for the timing generators. This process is iterated until the best match between the output transition and the reference edge is found.

As discussed earlier, there are actually four edges per waveform to calibrate (δ_1 through δ_4) in the pulsed modes. In order to calibrate each of these independently, the reference clock is run at one half the tester cycle rate and can be programmed to transition during either the even or odd cycle. Two different sets of calibration test

vectors are used to cause transitions in either cycle so that each δ can be separately calibrated. The RT mode presents a special problem in calibration, since there isn't a voltage transition on the output at the transition time unless the output is terminated to some intermediate voltage. Rather than require this external termination, the output is simply calibrated in another format, either RZ or RO, with the same timing parameters, and then the format is changed to RT. All paths through the formatter pass through two gate delays (Figure 4.9), so the edge timing is relatively independent of format selection and good RT calibration is achieved. Since the calibration procedure is performed per pin, the calibration of force channels can be performed with the DUT installed, which also compensates for mismatched driver performance due to unequal pin loading.

Calibration of the Acquire section of the channel is performed in a manner similar to the Force section, but since it is an input device, an indirect method must be used. First the Force side of the channel is programmed to transition at the desired sample time, then the sample timing generator is adjusted by the host until the transition is found. This procedure compensates for skews introduced by the timing generator as well as for any R-C effect of the input sample-and-hold.

Calibration software written for the experimental pin electronics chip was able to perform the calibration operation at the rate of about one channel per second. This could be improved significantly with better calibration algorithms and hardware assistance in the data transfer.

4.6 Results and Summary

This chapter has shown that the implementation of high-speed, high-accuracy pin electronics is possible even in a relatively low-speed base technology, such as CMOS. High resolution, low accuracy delay circuits, combined with closed loop calibration,

provide an area-efficient means of attaining accurate per-pin timing generation. Analog CMOS interface circuits, such as drivers, receivers, and comparators, have also been shown.

An experimental four-channel pin electronics chip which demonstrates these ideas was fabricated in a 2μ , double-metal CMOS technology. This device contained 13K transistors in a die size of 3.9 mm x 5.3 mm. The layout area required for each channel was $500\mu \times 2000\mu$. Running at a maximum frequency of 33 MVectors/sec, the chip dissipated 125 mW with a 5V supply. Calibration routines were written for the device and were successful in calibrating all edges to better than 1 ns over a wide range of delays and cycle frequencies.

The following chapter deals with the issue of vector storage. Various compression approaches are investigated to reduce the die area required by the vector memory, so that the pin electronics and vector storage sub-systems can be integrated onto the same chip.

Chapter 5

Vector Storage

5.1 Introduction

The purpose of the vector storage system is to supply the data necessary to drive the Device Under Test pins. A bi-directional pin driver needs four bits of vector information to specify all possible combinations of DUT driving and sensing in each cycle: a force value, a drive/inhibit bit to indicate direction, an expect value, and a mask bit to indicate if the data sensed by the pin is relevant. Some additional data may also be stored along with the vector that indicates what set of timing values or formats should be applied with the vector. Thus, the raw vector width is at least four times as wide as the maximum number of pins to be tested. For a multi-hundred pin tester, storing such a wide data format in external RAMS would necessitate a large number of components and would present considerable bandwidth difficulties as the design speed of the tester increased. The alternative of putting raw vector storage on-chip limits the maximum length of a test sequence that can be run due to the limited space for the on-chip RAM. The solution is to reduce the space consumed by the vector data through compression.

Compression can be applied to the vectors in various ways. One method is to

combine the semantics of two or more bits of the vector into a single bit. This reduces the tester flexibility somewhat but allows significant storage savings. This technique can, for instance, be applied to the force and expect data values. With both values in the same vector, a tester is able to expect a different value than it forced within a single tester cycle. While this feature is useful in cases where the drive value is tri-stated in the middle of a cycle, the cost of storing two data values seems high compared with the frequency of use of such a feature. The “redundant” data bit can be eliminated by storing a single data value and simulating the mid-cycle data value change by using two successive tester cycles.

A second way of achieving data compression is through the use of mapping. By adding a few bits to the overall test vector for each cycle, an entry in a map table can be indexed to produce additional data for the vector. This method is useful for reducing the storage requirements for the control bits such as inhibit and mask. Within a set of test vectors, inhibit and mask tend to operate uniformly on entire busses rather than on individual bits, typically resulting in a small number of unique patterns. By storing these patterns in a map table, a small number of bits can be used in each test vector to produce all of that vector’s control information, resulting in considerable savings in vector storage space (Figure 5.1).

Through the use of bit combination and mapping operations, the amount of storage required per DUT pin can be reduced from four bits to only one and a fraction bits. However, a considerable amount of redundant information still exists in the vectors. Repeated vector sequences and commonly occurring vectors carry less information than the bits used to represent them. In order to take advantage of this, a more general data compression method must be applied to eliminate the remaining redundant information. The remainder of this chapter is concerned with the identification and selection of a suitable compression method.

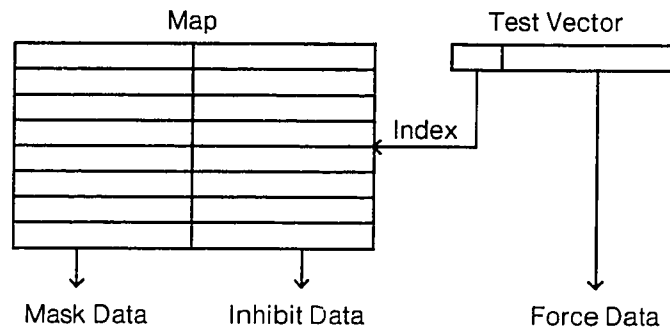


Figure 5.1: Data compression through mapping

5.2 Compression Alternatives

Selecting a fully optimal coding strategy for test vector compression is not possible because of the arbitrary nature of the input data stream. It is possible, however, to arrive at a good selection which will perform well under a wide variety of conditions. The first step in making such a selection is to consider the taxonomy of compression alternatives shown in Figure 5.2. As seen in the diagram, the initial classification of compression algorithms distinguishes the special-purpose algorithms from the general-purpose ones. Special-purpose schemes make use of *a priori* knowledge of the characteristics of the data stream in order to obtain good compression on certain classes of input. An example of a special-purpose situation would be a run-length encoder applied to the binary image data of a printed page. It is well known that individual scan lines of the image containing mostly uninterrupted background occur frequently in the data stream for the page. Thus, good results are obtained when a run-length encoder is applied. For test vectors, several examples of existing systems can be found where knowledge of the source data is used to obtain very high compression results. Memory testers are a prime example. In these devices,

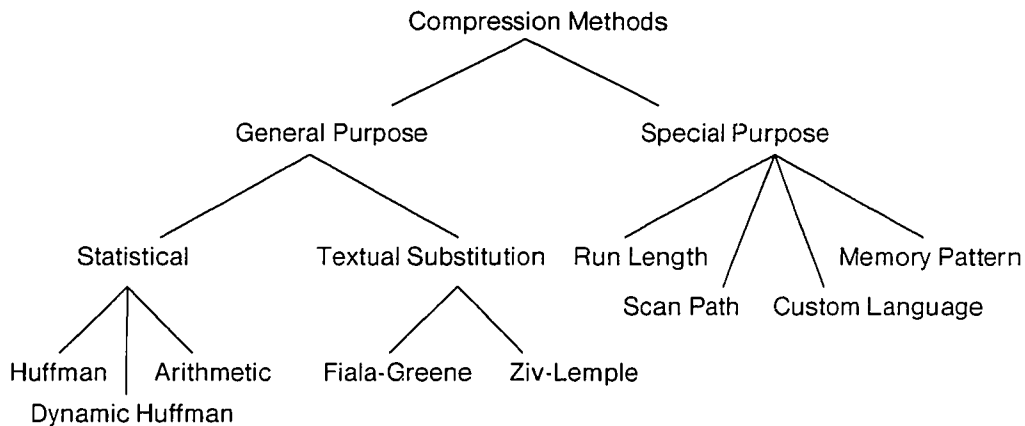


Figure 5.2: Taxonomy of compression alternatives

detailed knowledge of the data pattern to be applied to the test device is used in the design of a hardware engine for producing the desired vectors. Since the core of the vector generation algorithm is captured in the microcode of the engine, a single compressed “vector” can produce a long sequence of operations. Another example of a special-purpose compressor is a scan-path pattern generator [A183]. In scan-path designs, a large part of the internal state of the test device can be read or written by putting the device in a special mode that serializes all of the internal flip-flops. The state of the device is read or written by freezing the other device inputs and clocking the scan-path a single bit at a time. Since the scan-path is used repeatedly during testing to partition internal logic blocks, providing special hardware to perform the serial to parallel conversion of the scan path data (or alternatively a parallel to serial conversion of the vector data) results in a single vector being used for many clock cycles.

Compression methods such as run-length and scan path that rely on specific knowledge of the source data do very well for their intended tasks, but are restricted in the

types of devices that can be tested. A second, less specific, category of special-purpose compression schemes is the use of custom languages for compact representation. In keeping with the printed page analogy described above, a page description language such as Interpress [Xe86], can be used to specify the contents of a printed page. Such a high-level representation would yield a significant reduction in data beyond either the page bit-map or the run-length encoding. This reduction occurs because the structure of the data is captured at the source level, rather than in its compiled form. It seems feasible to design a vector description language using a set of primitive operators, such as clocks, counters, shifters, and so on. A language of this nature could have all of the expressive power of a conventional programming language, allowing an extremely efficient representation of the “object code” vectors. The burden of such a language, however, falls on the implementation of the decompressor. A single chip solution seems infeasible.

A more universal approach to vector compression is to employ general-purpose compression methods. While such schemes may not achieve the compaction efficiency that is obtained with special-purpose compressors on specific tasks, their wider applicability makes them more attractive. General-purpose methods can be partitioned into two classes: statistical and textual substitution methods. Statistical methods obtain compression efficiency, in general, by analyzing the *amount of information*, in an information theory sense, contained in source data. They then reduce the source to a minimal representation of the same data without loss of information. Textual substitution methods, on the other hand, use a *linear history* of the un-encoded source stream to adapt to the context of the source. Using an index to extract segments of the history stream, they obtain compression efficiency by re-utilizing portions of the un-encoded data. To more fully understand these alternatives, a basic example of each method is discussed in detail below.

Perhaps the purest example of a statistical method is Huffman coding [Hu52].

Huffman coding substitutes *codewords* for *symbols* (vectors) from the uncompressed source such that the bit length of the codeword for each symbol is inversely related to the frequency of occurrence of that symbol in the data stream. Frequently occurring symbols have very short codewords, while infrequently occurring symbols have longer ones. This technique results in a compression factor which is limited by the *entropy* of the source. The entropy of a data stream is defined by the sum:

$$H_0 = - \sum_{i=0}^{n-1} P(x = c_i) \log_2 P(x = c_i),$$

where x is a random symbol from the source and c_i ranges over all of the symbols of the source alphabet. Each coefficient:

$$- \log_2 P(x = c_i),$$

represents the number of bits necessary to encode the symbol c_i , while the weighted term:

$$-P(x = c_i) \log_2 P(x = c_i),$$

represents the contribution of c_i to the total number of bits, on average, of an optimal codeword. Thus the entropy of a set of symbols represents the minimum average number of bits necessary to represent each source symbol. The optimal compression ratio then is obtained by:

$$\text{compression ratio} = \frac{\text{symbol length}}{\text{entropy}}.$$

The average length of the codewords assigned in a Huffman encoding is usually only an approximation to the average length determined by the entropy. This is due to the fact that the codewords must be represented by an integer number of bits, while the entropy is a real number. Also, the decompressor implementation may limit the maximum length of a codeword, forcing a slightly sub-optimal codeword selection. The details of the algorithm for generating codeword assignments are not discussed

here, but are clearly illustrated by Knuth [Kn68]. The problem of compression loss due to fractional bits has been solved by a technique known as Arithmetic coding [Pa76]. This technique, however, results in only an asymptotic improvement in the overall compression while substantially increasing the complexity of both the compressor and decompressor.

When computing the overall compression ratio for a Huffman compressed stream, the size of the codebook, which contains the codeword-symbol pairs, is normally counted as part of the compressed data. In the VLSI implementation of the decompressor, the codebook resides in a special memory which performs the codeword to symbol association. Since this memory is distinct from the compressed data memory, the actual compression ratio can be computed solely on the basis of the space occupied by the codewords. Thus, ratios very close to that predicted by the entropy can be obtained. The downside of having this separate memory is that the codebook can exceed the available space, thus imposing an additional limitation on the maximum number of vectors that can be compressed.

The Huffman algorithm approaches an efficiency determined by the theoretical limit of the entropy. The entropy discussed so far, however, is only the zeroth-order entropy. In computing the entropy, each symbol of the input is considered in isolation; there is no memory of the ordering of the characters in the source. Greater compression efficiencies (lower entropies) can be achieved by considering the time-ordering of the input symbols in addition to their frequencies of occurrence. The first-order entropy, for example, is given by:

$$H_1 = - \sum_{i,j=0}^{n-1} P(x = c_i) P(y = c_j | x = c_i) \log_2 P(y = c_j | x = c_i).$$

Here, xy is a randomly chosen pair of adjacent symbols in the source stream. If repeated occurrences of adjacent symbols exist in the stream, then the first-order entropy will be lower than the zeroth-order entropy of the stream and better compression

is possible. Huffman algorithms which take advantage of higher-order entropies are complex [Kn85]. A much simpler way to take advantage of higher-order coherency is to employ a different class of general purpose compression method, the textual substitution compressor.

The key idea behind the textual substitution compression algorithm is to encode output stream segments as references to previously occurring segments. An additional mechanism allows for the inclusion of literal stream data when the segment to be output has not occurred in the recent past. Ziv and Lempel [ZL77] have shown that this technique approaches the efficiency bounds of the fixed codebook schemes, such as Huffman, which have full *a priori* knowledge of the source. Recently, Fiala and Greene [FG88] have described several methods for improving the original encoding of Ziv and Lempel by employing separate commands for the Copy and Literal operations and by making the format of these commands adaptive to the context of the source. They also describe a scheme (A1) which is explicitly tailored for simplicity and ease of decompression.

5.3 Compression Requirements

The number of proposed data compression schemes is large, which could potentially make the selection of a suitable algorithm a difficult problem. Fortunately, the physical constraints imposed by the need to implement the decompressor in a small amount of hardware, in addition to the system level requirements, narrows the search space considerably. The criteria used for evaluation of candidate compression/decompression schemes can be divided into three major categories: architectural implications, bandwidth considerations, and compression efficiency.

5.3.1 Architectural implications

It is assumed that the compression operation will take place on a general purpose host computer system, so the architectural implications deal only with the decompressor and the inherent qualities of the compressed data. The main requirements of the system architecture are:

Good compression relative to the decompressor die area

This encompasses a number of different compressor requirements. For minimal die area, both the internal storage requirements and the control complexity of the decompressor must be minimized. This implies a narrow data path, free of excessive pipelining or buffering. Control is reduced by limiting the number of codewords and fixing the size and position of the command and control fields. In general, since each gate in the decompressor takes away die area otherwise available for raw vector storage, the cost had better be justified.

Minimal critical resource restrictions in the decompressor

Critical resources are items such as RAM space or FIFO depth in the decompressor which may be exceeded during the course of the decompression operation. Ideally the only critical resource in the decompressor is the compressed vector memory itself. Having additional critical resources can lead to pathological cases where vector depth is limited by a resource other than the vector memory.

Limited branch capability

Performing either conditional or unconditional jumps within the test vector sequence is frequently desirable. The compression scheme should allow branches to predetermined target locations. It is acceptable for some local compression degradation to occur in the region of the target as a result of the target insertion.

5.3.2 Bandwidth considerations

The bandwidth considerations of the system deal the decompressor's ability to deliver a minimum decompressed output vector rate given the limited input bandwidth of the compressed vector storage RAM. If the decompressor is treated as a black box, the only requirements are on the decompressor input and output:

Limited number of decompressor input bytes required per cycle

The length of the input stream consumed by the decompressor within a single clock cycle must have a fixed maximum length. This determines an upper bound for the number of compressed data words which must be prefetched to execute any decompression operation within a single clock cycle.

Minimum decompressed output rate

The decompressor must guarantee that at least one decompressed data word is produced during each clock cycle. This removes any restrictions on local minimum compression rates and obviates the need for any sort of data FIFO. It also allows the decompressor clock to be the same as the tester cycle clock.

5.3.3 Compression efficiency

The compression efficiency of the system is the most subjective of the system requirements. Compression methods are often tuned for a particular style of input data and can perform very poorly on "unfamiliar" data. For test vectors, certain patterns are expected to occur frequently, such as repeated loops and fixed pins. The compression scheme should take advantage of these situations and produce high compression rates without prior knowledge of the source code. The compressor should adapt quickly to changes in the character of the input stream. In the worst case situation of random input data, the compression ratio should not be much less than one. These requirements are summed up as follows:

Good maximum compression for steady state inputs

Test vectors are often repeated many times. The compression algorithm should take advantage of this and produce asymptotically high compression rates for steady state inputs.

Good compression across the pins

Some pins may remain idle for many cycles while others within the same vector transition frequently. The performance of the compressor should be closely related to the number of active pins. This allows an automatic tradeoff between vector width and depth.

No *a priori* knowledge of the source code

Some compression algorithms use a static analysis of the source to determine fixed data which must be preloaded into the decompressor. This is undesirable, because of critical resources and the lack of ability to adapt to source context changes.

Adaptive

For a compression algorithm to be adaptive, good results must be achieved on a wide variety of source data contexts. This includes commonly occurring patterns such as clocks, counters, shifting bits, etc. The compression algorithm should also quickly adapt to frequent contextual changes in the input stream.

Degrades gracefully for random input data

The overhead of the encoding scheme should be such that the result of compressing a highly random data stream is not significantly larger than the original stream. Pathological worst cases should not exist.

5.4 Model Selection

Given the taxonomy of compression methods and the requirements for a good vector compression/decompression scheme described above, the search space of compression methods can be reduced considerably. Most of the special purpose methods can be dismissed quickly either because they are too problem-specific or because they have pathological cases which result in very poor performance. The one special purpose method which does seem somewhat attractive is the idea of a custom, high-level test vector language, but there are two basic problems with this approach. First, knowledge of the expected vector patterns must be applied in the selection of the set of primitives. This biases the compression scheme in favor of certain types of data. This limitation can be reduced by increasing the size of the primitive set. However, the richer the set is made, the more complex the decompressor must become. The second, perhaps more difficult, problem is that of source capture. In order to obtain vectors encoded in the source language, either a compressor must be devised, which can transform existing ones and zeros vectors into the source language (a de-compiler), or a new programming environment must be built to allow future patterns to be coded in the source language. Each of these problems represents a substantial technical challenge well beyond the scope of this work. Furthermore, even if such options existed, their solution would likely be impractical if a simpler compression scheme could be identified.

In the realm of general purpose compression methods, it was not possible from an abstract model of the statistical and textual substitution methods to determine which represented a better solution to the compression problem. It was therefore decided to examine the simplest instance of each method in more detail to determine the relative tradeoffs between the two. The statistical method implementation selected for study was the basic Huffman encoder. This model is the closest to the entropy theory upon

which all statistical encoders are based. For the textual substitution method the A1 encoding proposed by Fiala and Greene (FG) was selected, as this represented a slight performance improvement over the original Ziv-Lempel textual substitution model with no additional complexity. The comparison between the two methods was based on an analysis of the relative compression efficiencies of the encoding and also on the relative merits of the hardware implementation of the decompressors.

5.5 Decompressor Architectures

5.5.1 Huffman decompressor

A static Huffman encoder takes a set of source symbols of fixed size and maps them into a set of variable length codewords. The compressed stream is a concatenation of these codewords which have been regrouped into words whose fixed length is dictated by the wordsize of the system. In order to parse a stream of this nature, a compressed Huffman decoder must be able to re-tokenize the codewords (across machine word boundaries) and map these back into the original source symbols. A block diagram of an architecture for performing such a transformation is shown in Figure 5.3.

The decoder consists of four main sections: vector storage, prefetch buffer, codeword map, and control. The vector storage is a random access memory which holds the entire compressed data stream. The width of the memory must be at least as great as the longest anticipated codeword in order to meet the requirement of a minimum decompressed output rate of one vector per decompressor cycle. The depth of the memory is limited only by the available chip real estate. The data delivered by the vector RAM is held in the prefetch buffer. The prefetch buffer ensures that two successive compressed words are always available to the shifting network so that the next complete codeword can be extracted, even if it crosses word boundaries. The control section keeps track of the position of the compressed data within the prefetch buffer

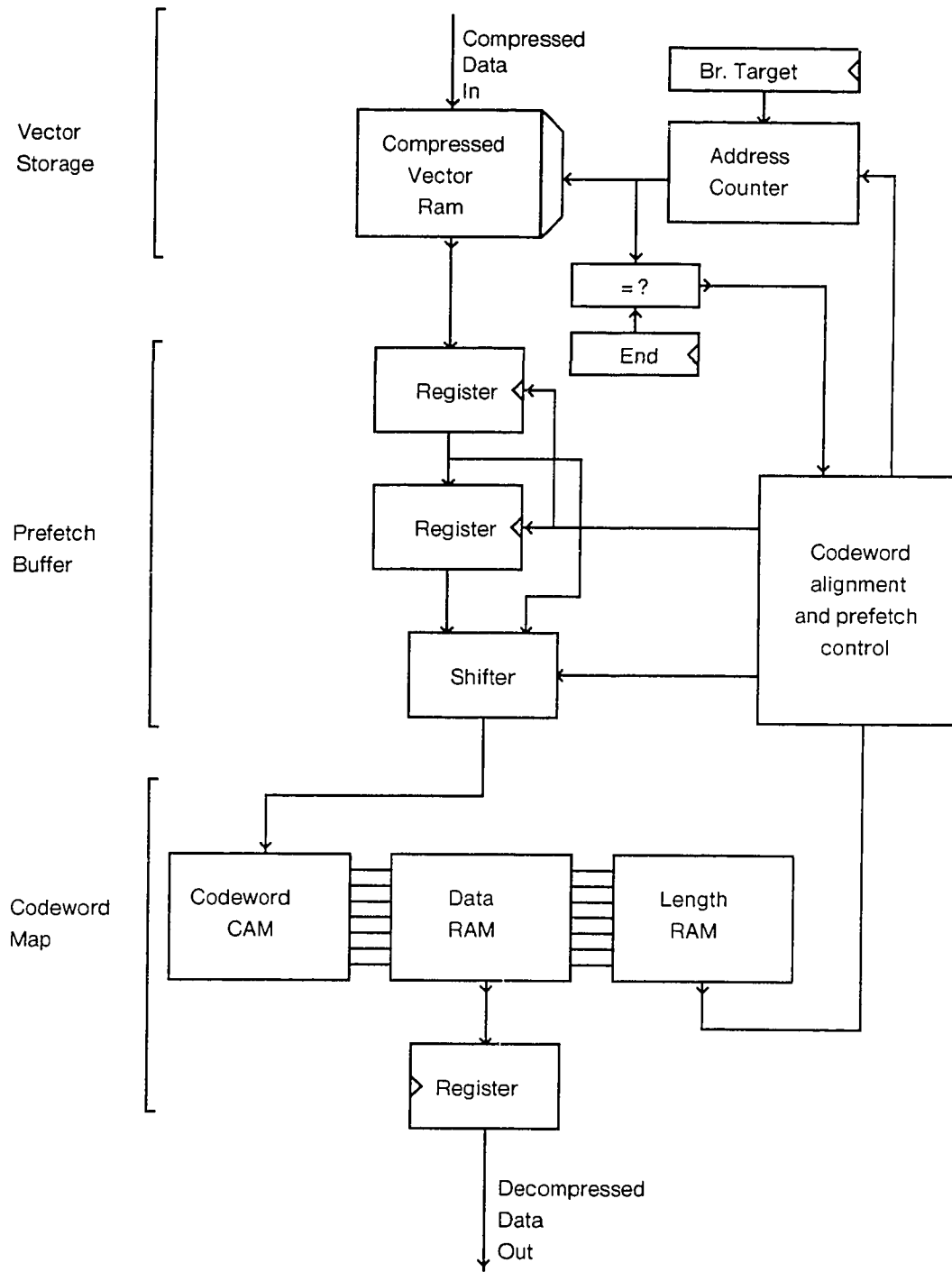


Figure 5.3: Huffman decoder block diagram

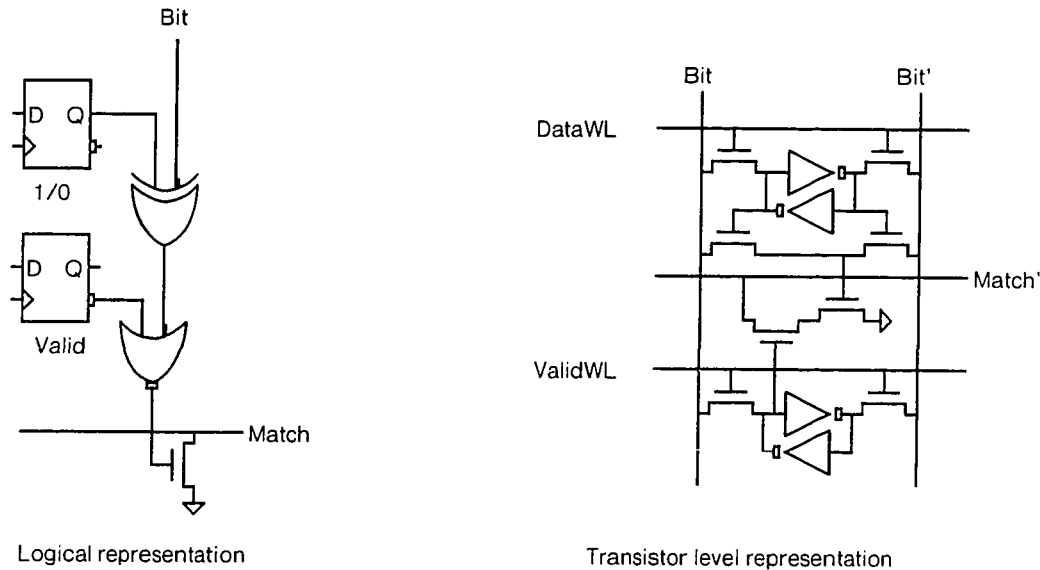


Figure 5.4: Codeword CAM cell schematic

pipeline and also takes into account the length of the current codeword. With this information the bit address of the first bit of the succeeding codeword can be determined; it is then extracted and left-justified by the shifter. The output of the shifter, then, is a concatenation of the next codeword, contained in the most-significant bits, with succeeding codeword or codewords in the low order bits. The purpose of the codeword Content-Addressable Memory (CAM) is to match the unique codeword prefix extracted by the shifter and ignore the remainder of the data. To implement this function, each cell of the codeword CAM consists of two bits of storage (Figure 5.4). One of these bits stores a value that the cell uses to match with the shifter output data, while the other bit encodes whether or not the CAM cell should participate in the matching of the codeword at all. In operation, each line of the CAM is loaded with a different codeword prefix in the most significant bits of the CAM. If the codeword held in a line is shorter than the maximum length codeword, then the low order bits

of the line will have their Valid bits cleared; the rest are asserted. In the first half of the cycle, the Match line is precharged high, while the Bit and Bit' lines are discharged to ground. Once precharge has completed, the Bit and Bit' lines are driven with the codeword data extracted by the prefetcher. Each CAM cell then performs the comparison operation of its bit. If the cell does not match and the Valid bit is set, then the Match line is pulled low. After all the compare operations have completed, one Match line, corresponding to the unique prefix code applied to the inputs, will remain high. This one high Match line is then used as a word select line in the Data and Length RAMS. The Data RAM supplies the decompressed output symbol which becomes the next word of the decompressed output stream. The Length RAM informs the control logic of the number of bits which were actually contained in the codeword. This allows the control logic to compute the bit-address of the most significant bit of the next codeword so that the shifter can be set up for the next decompressor cycle. The length information contained in the Length RAM is statically loaded at the same time that the codewords and symbols are loaded in the CAM and Data RAM.

5.5.2 Fiala-Greene decompressor

The FG encoding takes a set of source symbols of variable length and maps them into a set of variable length codewords. The compressed FG stream consists of a sequence of command words interspersed with literal data words. Two types of commands are used: Copy and Literal. The Copy command has two parameters: the length of the run of symbols to be copied and their position in the history buffer. The Literal command needs only one field: the length of the run of characters which follow in the compressed stream. When Copy is interpreted, a sequence of source symbols corresponding to the number specified by the length field is transcribed from the history buffer of the decompressor to the output stream. The index of the first character in the buffer is indicated by the position field. At the end of each

decompressor cycle, the current output symbol of the decompressor is copied into the least recently written location of the history buffer so that the buffer always contains a history of the most recently decompressed data. The block diagram of an architecture which implements these functions is shown in Figure 5.5. The vector storage RAM and prefetch buffer are nearly the same as those described for the Huffman decoder. The principal difference is that the shifter needed to extract the command word from the prefetch buffer has been replaced by a pair of multiplexors. In the Huffman scheme, codewords contain a variable number of bits, which necessitates a shifter which operates at the bit level. The boundaries between codewords and literal data in the FG scheme are all at the word level which makes the selection of the next operand somewhat simpler. In order to meet the minimum data delivery requirement of one word per clock cycle, the RAM word width must be at least as great as the number of words required for a single Literal command and one word of literal data. Since the Copy and Literal command words are the same length, this guarantees that Copy commands can meet the minimum data delivery requirement as well, since only the command itself is needed to produce an output symbol. There are two outputs of the prefetcher in the FG scheme: the next command and a literal data word. The command is interpreted by the control logic to determine whether the next decompressed output word should come from the history buffer or the literal data stream. The multiplexor at the output of the history buffer selects between these two alternatives. Unlike Huffman, the control logic in the FG decoder can determine, merely by decoding the current instruction, where the next instruction occurs in the prefetch buffer. This obviates the need for the RAM length field in the history buffer.

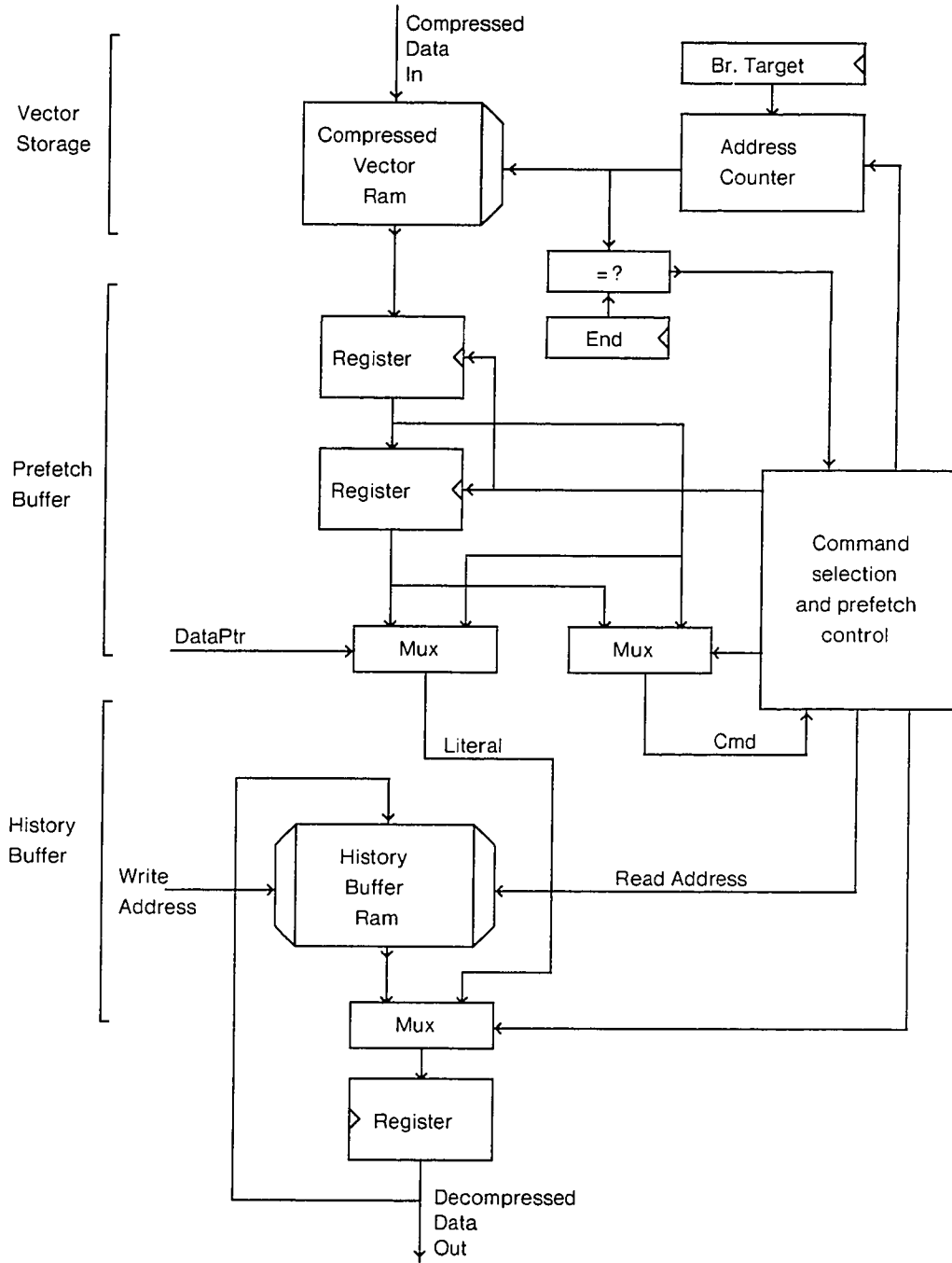


Figure 5.5: Fiala-Greene decoder block diagram

5.6 Compression Efficiencies

Before the analysis of the two compressor/decompressor models could be performed, it was first necessary to identify the parameters for the decompressed data format and also to select a set of benchmark test vectors. The format required by the decompressor in the actual tester chip consists of a four bit control field and a sixteen bit data field (this is described in more detail in Chapter 6). The control information determines which bits are inputs and which are outputs and also which bits participate in the input comparison operation. This data was not available for all of the sample vector files, so the comparisons were made only on the basis of the compressibility of the data portion of the vectors. It is expected that if the control data were included, the compression ratios would actually improve, since there is a high degree of correlation between the control and data fields of a vector. A suite of sixteen test vector files was obtained from industrial sources to benchmark the performance of the two compressor models. The vector files contained traces from three different chips, a floating point multiplier (FP), a micro-processor execution unit (EU), and a display controller chip (DC). These three chips represent three substantially different vector styles: the vectors from the floating point unit contain largely shifting data patterns used to test the multiplier, the execution unit vectors contain short repeated instruction sequences, and the display controller vectors contain mostly random logic test. The compression results for each of the three chips using a static Huffman coding are shown in Table 5.1.

As seen in the table, the number of unique codewords in the vector files varies greatly. EU0 requires a rather large number of entries (434) in the codeword table. If this value were to exceed the actual size of the table this could limit the total number of vectors that could be compressed. Another interesting point is that the entropy does not necessarily correlate with the number of codewords. This can be seen by

| Data-file | Vectors | Codewords | Entropy | Compression |
|-----------|---------|-----------|---------|-------------|
| FP0 | 2112 | 47 | 5.11 | 0.319 |
| FP1 | 2112 | 59 | 5.78 | 0.361 |
| EU0 | 4634 | 434 | 5.44 | 0.340 |
| EU1 | 4634 | 51 | 2.92 | 0.182 |
| EU2 | 4634 | 58 | 3.09 | 0.193 |
| EU3 | 4634 | 176 | 3.51 | 0.219 |
| EU4 | 4634 | 49 | 2.89 | 0.180 |
| EU5 | 4634 | 2 | 1.00 | 0.063 |
| DC0 | 1223 | 2 | 0.64 | 0.040 |
| DC1 | 1223 | 7 | 1.71 | 0.107 |
| DC2 | 1223 | 9 | 2.87 | 0.179 |
| DC3 | 1223 | 119 | 5.00 | 0.312 |
| DC4 | 1223 | 127 | 5.18 | 0.324 |
| DC5 | 1223 | 67 | 3.99 | 0.250 |
| DC6 | 1223 | 155 | 5.47 | 0.342 |
| DC7 | 1223 | 126 | 4.85 | 0.303 |

Table 5.1: Static Huffman coding

comparing the data from FP1 and EU0. FP1, which requires a rather small number of codewords, has the highest entropy of any of the vector sets, even EU0. This is due to the fact that while EU0 has many unique codewords, most of them occur very infrequently, so that the few which do occur often can be coded with relatively few bits.

In the Huffman encoder, there are no parameters that can be used to vary the compression efficiency. The codeword table is simply made as large as possible to avert overflow. This is not the case for the FG encoder. There are, in fact, several parameters which can be adjusted to maximize efficiency: the size of the history buffer, the size of the codewords, and the rules used by the compressor to select the encoding strategy. The size of the history buffer in the FG model has the most direct impact. Ideally, the larger the buffer the better the efficiency, but as more space is consumed by the buffer, less is available for the compressed vector memory storage.

Also, as the size of the buffer is increased, the field lengths in the compressor codewords must expand to accommodate the varying number of buffer address bits. This results in additional overhead in the codewords, which in turn reduces the overall compression. Thus, a point of diminishing returns exists where the average efficiency actually decreases as the buffer is made larger. A number of experiments were conducted using various parameters for the size of the instruction words and the length of the history buffer. For the sake of simplicity in the decompressor, the Copy and Literal command were assumed to always be coded in equal length words. Since the Copy command contains two parameter fields to the one of the Literal command, Copy determines the coded command word width. Table 5.2 shows the various buffer sizes used in the experiments and the corresponding field sizes.

| Buffer Size | Length Field | Position Field | Total Bits |
|-------------|--------------|----------------|------------|
| 32 | 3 | 5 | 8 |
| 64 | 4 | 6 | 10 |
| 128 | 4 | 7 | 11 |
| 256 | 4 | 8 | 12 |
| 512 | 4 | 9 | 13 |
| 1024 | 4 | 10 | 14 |

Table 5.2: Copy command parameters

The results of the compression experiments are shown in Table 5.3 and the two graphs, Figure 5.6 and Figure 5.7. As can be seen from the last column of the table, the maximum compression is not always achieved with the largest history buffer.

Table 5.4 shows the number of times each of the buffer sizes produced the optimum compression efficiency for the sixteen benchmark vector sets. As can be seen from these results, there does not seem to be a strong correlation between the buffer size and its compression efficiency. This is not too surprising, since the test vector data is basically binary data much like the BF data set used in the experiments by Fiala and Greene. In their results, they observed fairly little sensitivity of the compression

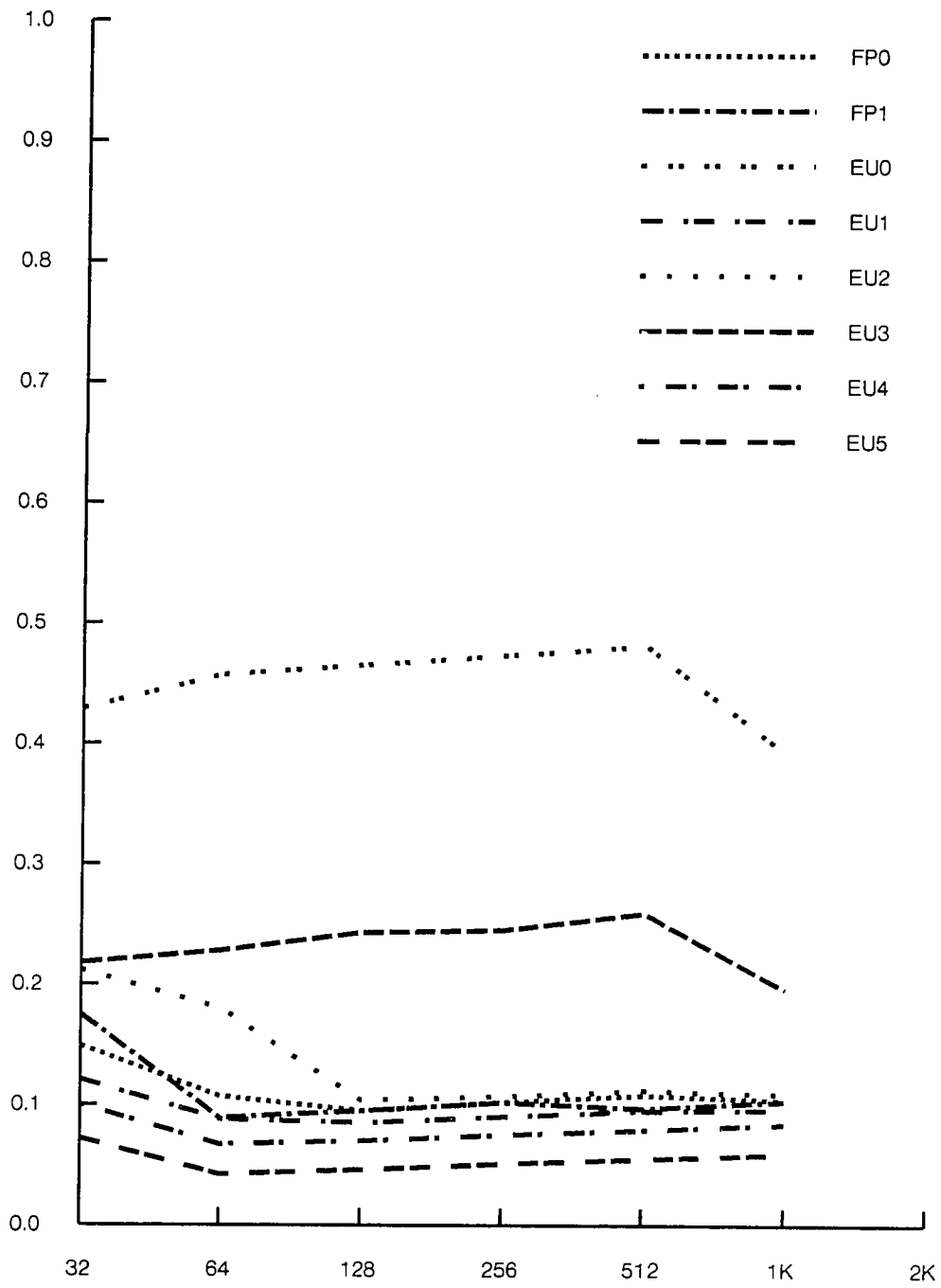


Figure 5.6: FG compression vs. buffer size: FP and EU

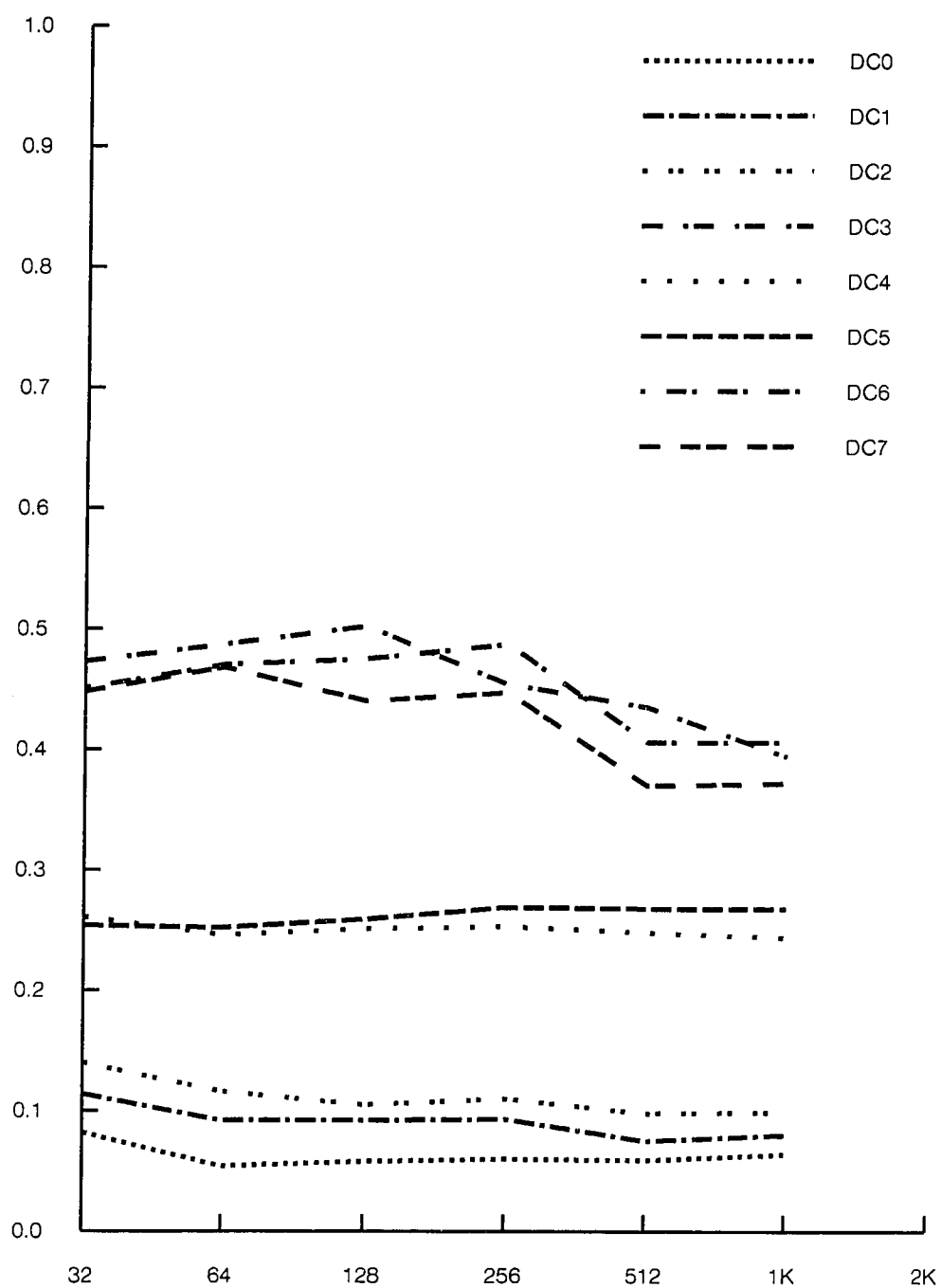


Figure 5.7: FG compression vs. buffer size: DC

| File | 32 | 64 | 128 | 256 | 512 | 1024 | Optimum |
|------|-------|-------|-------|-------|-------|-------|---------|
| FP0 | 0.149 | 0.107 | 0.095 | 0.102 | 0.108 | 0.104 | 128 |
| FP1 | 0.175 | 0.089 | 0.095 | 0.102 | 0.097 | 0.103 | 64 |
| EU0 | 0.429 | 0.457 | 0.465 | 0.473 | 0.481 | 0.392 | 1024 |
| EU1 | 0.121 | 0.088 | 0.085 | 0.090 | 0.095 | 0.096 | 128 |
| EU2 | 0.212 | 0.180 | 0.104 | 0.107 | 0.112 | 0.109 | 128 |
| EU3 | 0.218 | 0.228 | 0.243 | 0.245 | 0.259 | 0.196 | 1024 |
| EU4 | 0.100 | 0.067 | 0.070 | 0.075 | 0.079 | 0.084 | 64 |
| EU5 | 0.072 | 0.042 | 0.046 | 0.051 | 0.055 | 0.059 | 64 |
| DC0 | 0.082 | 0.054 | 0.058 | 0.060 | 0.059 | 0.064 | 64 |
| DC1 | 0.114 | 0.092 | 0.092 | 0.093 | 0.075 | 0.080 | 512 |
| DC2 | 0.140 | 0.116 | 0.105 | 0.110 | 0.098 | 0.099 | 512 |
| DC3 | 0.473 | 0.487 | 0.502 | 0.454 | 0.435 | 0.394 | 1024 |
| DC4 | 0.261 | 0.246 | 0.251 | 0.253 | 0.248 | 0.244 | 1024 |
| DC5 | 0.254 | 0.252 | 0.259 | 0.269 | 0.268 | 0.268 | 64 |
| DC6 | 0.451 | 0.470 | 0.475 | 0.487 | 0.406 | 0.406 | 512 |
| DC7 | 0.448 | 0.468 | 0.440 | 0.447 | 0.370 | 0.372 | 512 |

Table 5.3: FG experimental results

ratio to buffer size for a binary bootfile. The important point that can be extracted from these results is that buffer sizes as small as 64 entries can produce reasonable compression results.

| Buffer Size | Optimum Hits |
|-------------|--------------|
| 32 | 0 |
| 64 | 5 |
| 128 | 3 |
| 256 | 0 |
| 512 | 4 |
| 1024 | 4 |

Table 5.4: Number of optimum efficiency occurrences vs. buffer size

5.7 Analysis of Results

Based on the compression efficiency analysis and the details of the decompressor architectures, a one-to-one comparison of the Huffman and Fiala-Greene methods can be made using the requirements discussed in Section 5.3.

5.7.1 Architectural comparison

Good compression relative to the decompressor die area

The Huffman decompressor definitely suffers from the fact that the codebook must be made as large as possible to avert overflow problems. From the benchmark test cases, the desirable length appears to be at least several hundred entries. The FG history buffer, which is roughly analogous to the Huffman codebook, appears to work quite well even with as few as 64 entries. Furthermore, the Huffman entries are nearly twice as long, due to the CAM, and are substantially more complex. This severely limits the RAM size available to store the compressed vector data.

Minimal critical resource restrictions in the decompressor

Codebook limits in Huffman. None for FG.

Limited branch capability

Branching is feasible in both methods, although each requires some post processing of the compressed stream. To simplify the hardware in both architectures, it is desirable to have the codewords of the compressed data stream word-aligned at both the branch End address and the branch Target address (Figure 5.8). This serves two purposes. First, it guarantees that the codeword at the branch Target will be fully contained in a single fetched word so there are no problems associated with exceeding the data bandwidth of the vector RAM. Secondly,

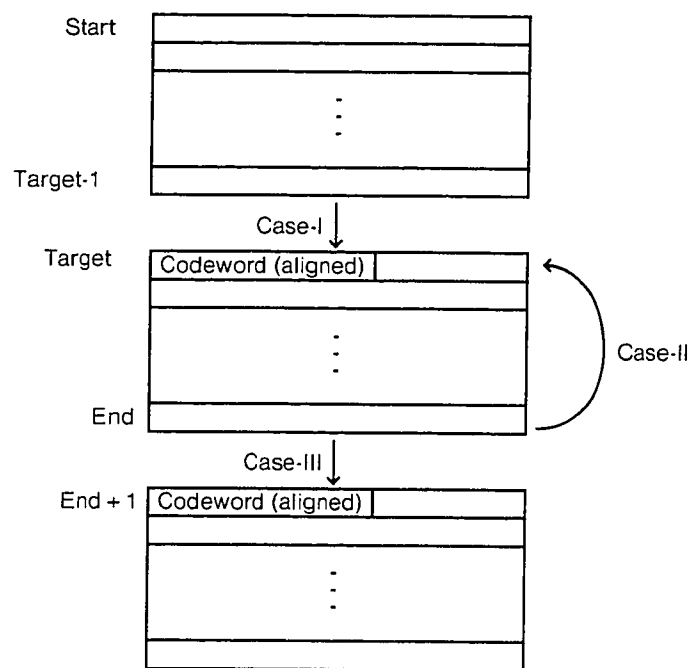


Figure 5.8: Branch operation

it guarantees that the decompressor sees a uniform data stream regardless of which path is taken through the Target and End areas. Figure 5.8 shows three cases. In Case-I, the decompressor is executing code in the block before the target address. Since the code stream is contiguous at the Target, no special action is needed by the decompressor when the Target address is crossed. When the decompressor reaches the End address and the branch is taken, Case-II, it has exhausted all of the bits at the End address (since End+1 is by definition a new codeword). A fetch of the data at the Target address will merge neatly into the compressor data stream. In the case where the branch is not taken, Case-III, the resulting stream is similar to Case-I.

This technique allows conditional branches of arbitrary length. However, a few subtleties have to be dealt with. In either Huffman or FG coding, a method is needed to implement word-alignment of codewords. This can be accomplished in Huffman by substituting existing codewords in the data stream with new codewords whose bit-length is based on the length of the required pad bits. The cost of this method is a few entries in the codebook and a slightly lower compression ratio. In FG coding, the padding need only be done on a byte basis, since there is no bit-level coding. To vary the byte-length of the coded stream, existing codewords are broken up into a sequence of less compact codewords. A Copy command of length three, for example, can be broken in three Copy commands of length one, thereby consuming three additional bytes in the stream. Literal commands can likewise be divided, if sufficient Copys are not present in the stream. An additional complexity encountered when branching in an FG code stream is that the state of the history buffer is different each time the block between Target and End is executed. The compressor must therefore assume that the content of the history buffer is invalid, i.e., it may not issue

any Copy commands at the start of each of the three blocks (pre-Target, Target to End, post-End). At worst (all literal data needed to refill buffer), this consumes space in the compressed stream that is three times the history buffer size to reestablish the state of the buffer.

5.7.2 Bandwidth comparison

Limited number of decompressor input bytes required per cycle

Both of the alternative compression methods do well. Both methods can limit the maximum codeword length, although Huffman has a slight compression loss.

Minimum decompressed output rate

Each of the architectures delivers one output vector per cycle. The FG scheme can be more heavily pipelined since the processing of the current instruction does not affect the fetching of the next. In Huffman, the path through the CAM, Length RAM, and shifter must be evaluated within a single clock cycle. This will be the limiting factor in Huffman performance.

5.7.3 Compression efficiency comparison

Good maximum compression for steady state inputs

Both of the alternative compression methods do well. For Huffman the maximum compression is the minimum codeword length divided by the output symbol length. For FG, it is the copy length divided by the length of the maximum number of copied words.

Good compression across the pins

Both of the alternative compression methods do well.

No *a priori* knowledge of the source code

True for FG, not true for Huffman.

Adaptive

The FG decompressor is by nature better at adapting to contextual changes in the source code because the contents of the history buffer are constantly changing. The Huffman method utilizes fixed data which has been pre-computed and downloaded into the decompressor. Methods exist for dynamically changing the Huffman codebook, but these add substantially to the complexity of the design.

Degrades gracefully for random input data

In the presence of random inputs, the FG decompressor does much better than the Huffman one. For long sequences of data that contain no repeated information, FG simply codes the string in terms of the maximum length literal operation. The worst-case compression, then, is the ratio of Literal command plus data to data alone. Since the maximum literal length is quite long (typically 64 output words), the value for the worst-case compression is very near one. Huffman, on the other hand, is limited by the size of the codebook CAM/RAM, so the compression ratio is undefined.

5.8 Conclusions

Starting with an initial four vector bits per DUT pin, it has been shown that the control and data fields of a vector can be packed into slightly more than one bit, on average, with little impact on the flexibility of the tester. Further reduction can be achieved through general data compression techniques. A number of compression methods have been discussed and two different compression methods, Huffman and Fiala-Greene, were examined in greater detail for their applicability to the problem.

The compression results for both methods looks promising. The average compression achieved over a suite of test vectors using the Huffman method was 4.3 : 1, while the average for the FG method was 4.7 : 1 (taken across all buffer sizes). The main disadvantages of the Huffman code are the size of the codebook *CAM/RAM/* and the poor compression performance for random input data. The real test of the feasibility of using compression is to produce a full decompressor implementation and determine the relative space efficiency of utilizing the tester die area for additional storage capacity versus using it for decompression logic. The next chapter presents the design of a complete integrated tester chip which employs some of the compression concepts discussed here.

Chapter 6

Implementation

6.1 Introduction

The preceding chapters have described the fundamental ideas for building integrated tester systems. Chapter 3 discussed the basic system architecture and requirements, while Chapters 4 and 5 covered in some detail the problems of pin drive and vector storage. This chapter brings together these ideas in the design of *Testarossa*, a complete integrated tester chip. Testarossa implements all of the functions of pin electronics, vector storage, decompression, error capture, and test control for sixteen DUT channels. By integrating all of these tester functions on a single piece of silicon, it is possible to build extremely compact testers, thereby eliminating many of the electrical problems associated with larger test systems.

6.2 System Overview

The goal in designing Testarossa was to produce a single device that could be used as a building block for constructing high speed, high pin-count, VLSI test systems. Figure 6.1 illustrates the configuration for a 256 pin test system using the Testarossa

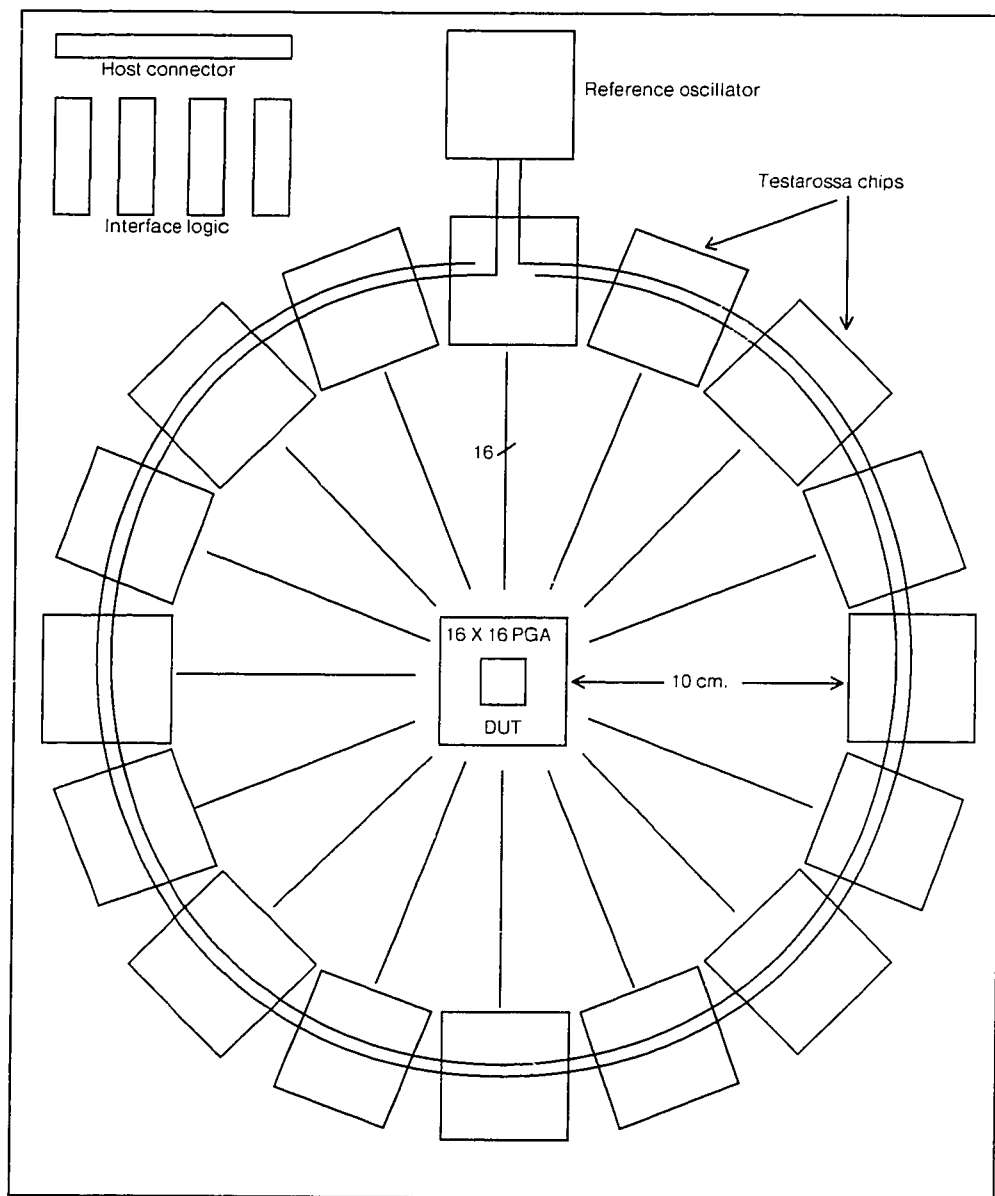


Figure 6.1: Single board integrated test system

chip. The system consists of sixteen tester chips, each providing sixteen I/O channels. The chips are arranged in a circular fashion around the central DUT, thereby equalizing the lead lengths, while at the same time maintaining a total trace length of less than ten centimeters. This limits the time-of-flight of a signal between the DUT and the pin electronics to only a few hundred picoseconds, obviating the need for terminated transmission lines. The short trace length also provides a stray load capacitance on the DUT outputs on the order of only a few picofarads. This ensures high-fidelity waveform transmission for both driven and sensed DUT signals with a minimum of signal loading.

As illustrated in Figure 6.1, the number of support components necessary to bind the system together is small, consisting of a few transceivers to buffer the host signals and the necessary chip select logic for addressing individual tester chips. The only additional logic necessary is that required to generate the reference clock. The figure shows a single chip solution to generating this signal, though since only one copy of the generator is required, it could also be produced by a commercial pulse generator or a collection of standard components. The real difficulty with this signal is not its generation, but rather its uniform distribution to all of the tester chips with minimal skew. Minimal skew in the reference clock distribution network is essential since all timing calibration is performed relative to the reference clock input of each tester chip. The physical layout of the tester chips is optimized for the radial DUT signals which tends to make the distribution of such bussed signals difficult. There are, however, several solutions to this problem, and Figure 6.2 illustrates a particularly elegant one. By distributing signals differentially in a counter-rotating fashion, each chip in the loop can detect a transition with zero skew relative to other chips in the loop simply by receiving the two counter-rotating signals with a differential amplifier. By making the rise time of the differential signals slightly greater than the propagation time around the loop, the crossover point is the same independent of position on the

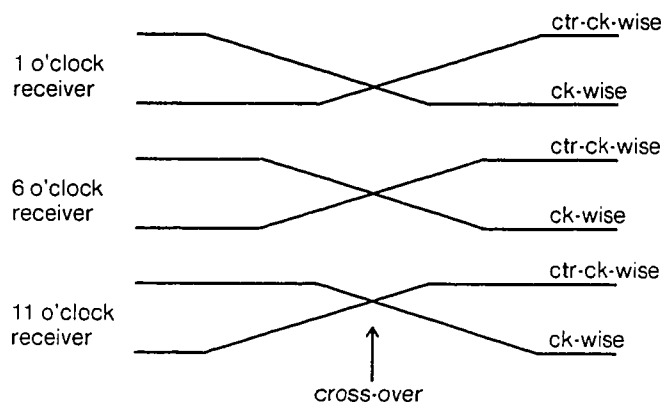


Figure 6.2: Skew elimination by counter propagating signals

loop [CC87]. The remainder of the bussed clock and control signals that connect the chips in the loop are neither skew nor delay sensitive and thus can be routed in the straightforward manner.

6.3 Testarossa Architecture

The Testarossa chip is composed of three main parts: the pin electronics, the decompressor and associated sequence control logic, and the vector storage RAM (Figure 6.3). The pin electronics section provides 16 I/O channels employing the timing generation, formatting, driving, and sampling techniques discussed in Chapter 4. Due to limited die area though, only one set of calibration and format selection registers was implemented.

The vector storage RAM was designed using a dynamic, self-timed, one transistor per cell, folded bit line RAM architecture¹. The 100 fF storage cell was constructed using an $11\mu \times 11\mu$ gate-oxide capacitor. This resulted in an overall cell size of $14\mu \times 20\mu$ with a 2μ technology. This RAM architecture allowed a $1K \times 40$ bit RAM

¹Many thanks to Russell Kao who did the simulation and layout of a prototype version of the DRAM.

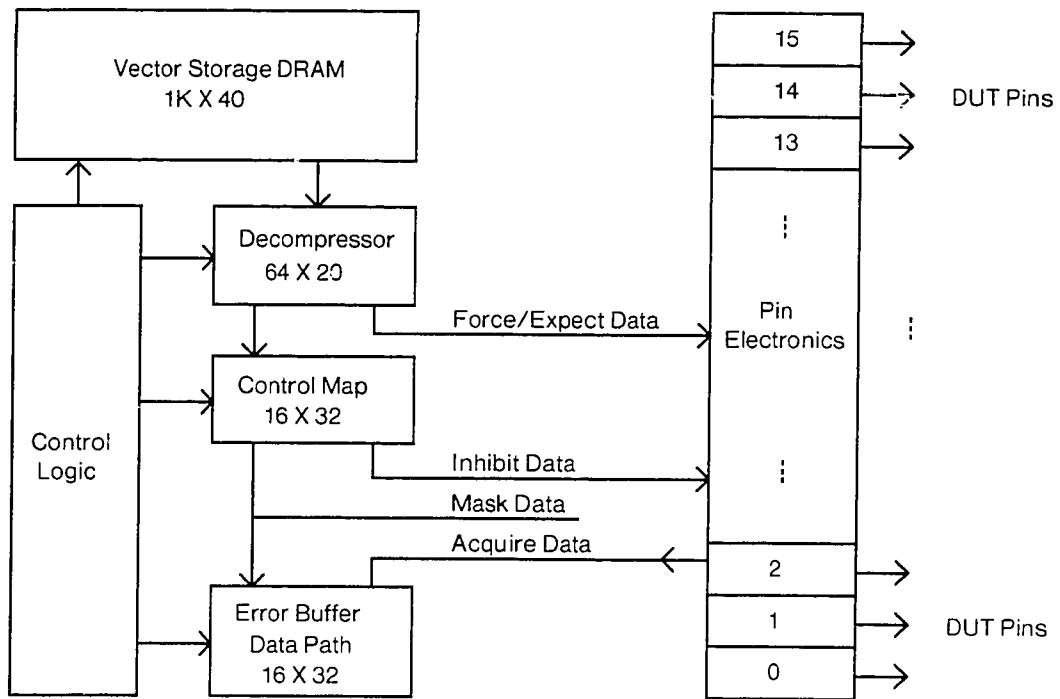


Figure 6.3: Block diagram of Testarossa chip

| Entry | 16 bits | 16 bits |
|-------|-------------------|----------------|
| 0 | Inhibit Vector 0 | Mask Vector 0 |
| 1 | Inhibit Vector 1 | Mask Vector 1 |
| 2 | Inhibit Vector 2 | Mask Vector 2 |
| | | |
| | | |
| | | |
| 15 | Inhibit Vector 15 | Mask Vector 15 |

Figure 6.4: Control map

to be implemented in a die area of only $3.0 \text{ mm} \times 5.6 \text{ mm}$ and provided a nominal decompressed vector storage capacity of 10K vectors. The selection of dynamic RAM for vector storage was a rather risky decision because of the noise sensitivity of the analog sense amplifier circuitry. However, since the dynamic design had been proven in a prototype chip, the advantage of the increased storage capacity seemed worth the risk, and dynamic storage was used.

Testarossa employs the Fiala-Greene compression technique and the control mapping method outlined in Chapter 5 as a means of reducing the on-chip storage requirements. The Fiala-Greene algorithm was selected on the basis of the good compression results obtained with relatively modest hardware requirements. With a history buffer size of only 64 entries, the compressor averages a 4.7 : 1 compression ratio over the sample vector suite. The mapping strategy is used to reduce the storage requirements for the mask and inhibit data. Since most DUT pins are either unidirectional or elements of wide bi-directional busses, only a moderate number of map entries are required. Each test vector can select from one of sixteen combinations of mask and inhibit bit patterns to be applied along with the force or expect data (Figure 6.4). A single test vector, then, consists of 20 bits. The low-order 16 bits of the vector are

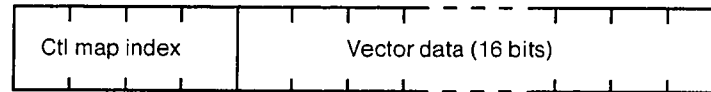


Figure 6.5: Test vector fields

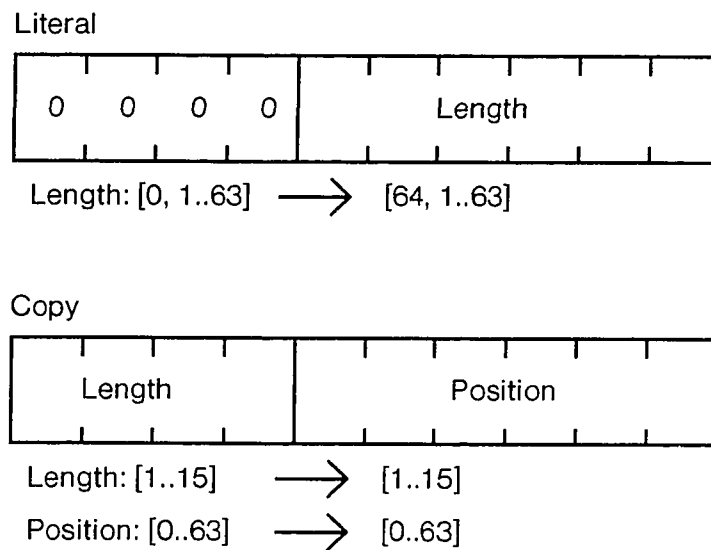


Figure 6.6: Compression opcodes

the force/expect data, while the high-order four bits form the index for the control map (Figure 6.5).

This 20 bit requirement for vectors manifests itself throughout the Testarossa architecture in a number of ways. The compressed data stream must efficiently pack 20 bit literal vector data words into the stream with little or no overhead, along with the opcodes that direct the decompression operation. The size of the history buffer and the length of the literal and copy operations require considerably less than 20 bits to express, so a sub-multiple length of 10 bits was chosen for the compression opcode size. Figure 6.6 shows the format of the opcodes implemented by the decompressor

and how the fields pack into a 10 bit “byte.” The first four bits of the opcode determine the command type: if the bits are all zero, then the command is a Literal, otherwise it is a Copy. The Literal command has only one field, the length field, which indicates the number of literal vectors which follow in the compressed stream. The Copy command has two fields: the copy length field, which indicates the number of vectors to include in the output stream from the History Buffer, and the position field, which points to the location in the History Buffer where the copy operation is to begin. Copies which wrap around from the end of the buffer back to the beginning are permitted.

Figure 6.7 shows the decompressor data path with all bus widths annotated. In addition to vector generation, the Testarossa data path also contains the Control Map and error detection and recording logic as shown in Figure 6.8. The data captured by the pin electronics is pipelined one stage, then compared with the Expect data along with the Mask information. If a mismatch occurs on a bit which is not masked, an error is signaled. This causes the failing data to be written into the Error Buffer along with the cycle number in which the error occurred. This information is then available to the host after completion of execution. The buffer pointer is incremented after each error to point to the next empty location. When the buffer becomes full, further errors are ignored so that only the initial sixteen errors are recorded. Additional errors can be captured by changing the contents of the control map to mask out earlier errors and re-running the test.

The following sections present a brief “data sheet” description of Testarossa. The device pinout, internal register model, and low-level programmers’ interface are described in detail.

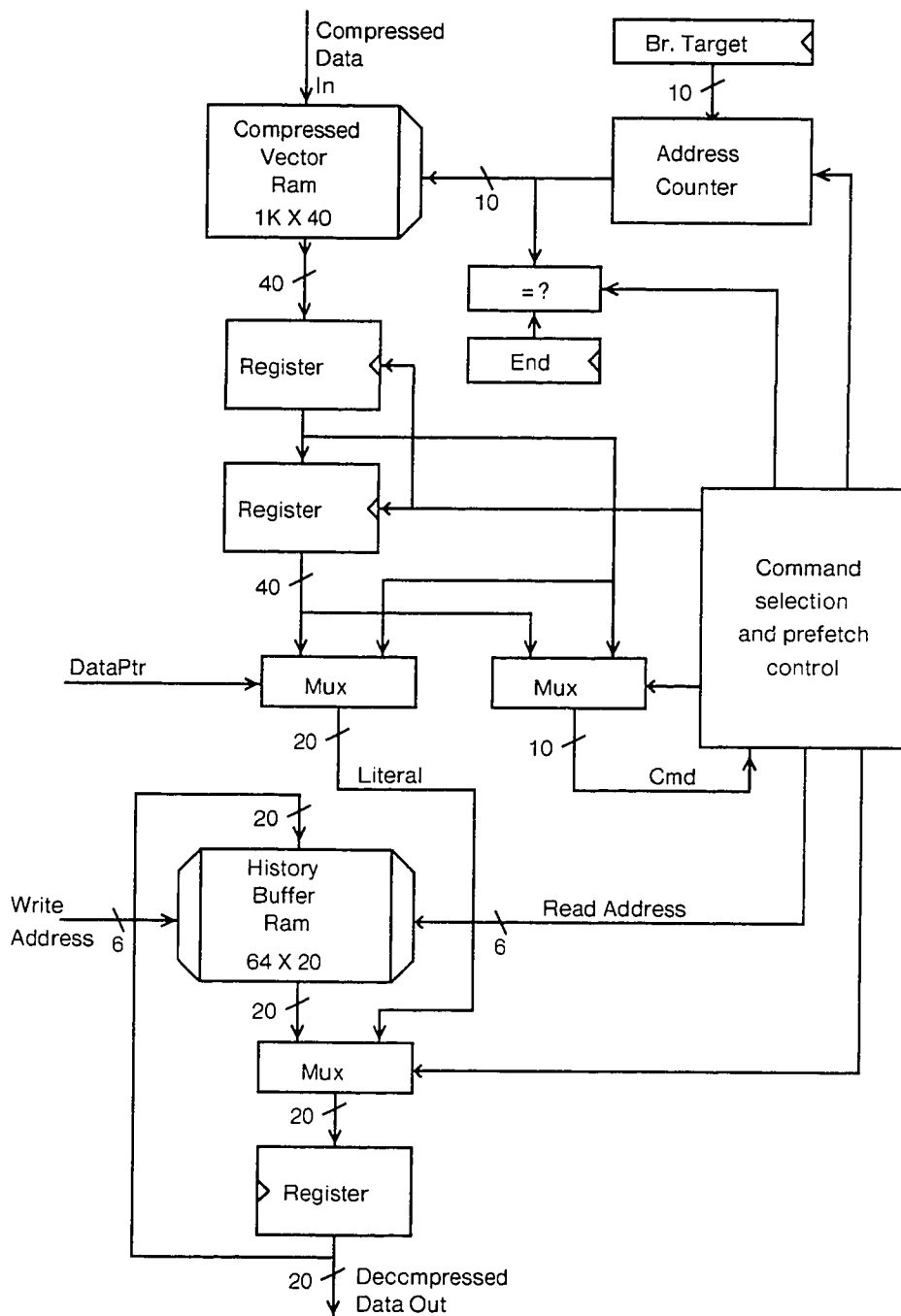


Figure 6.7: Decompressor data path with bus widths

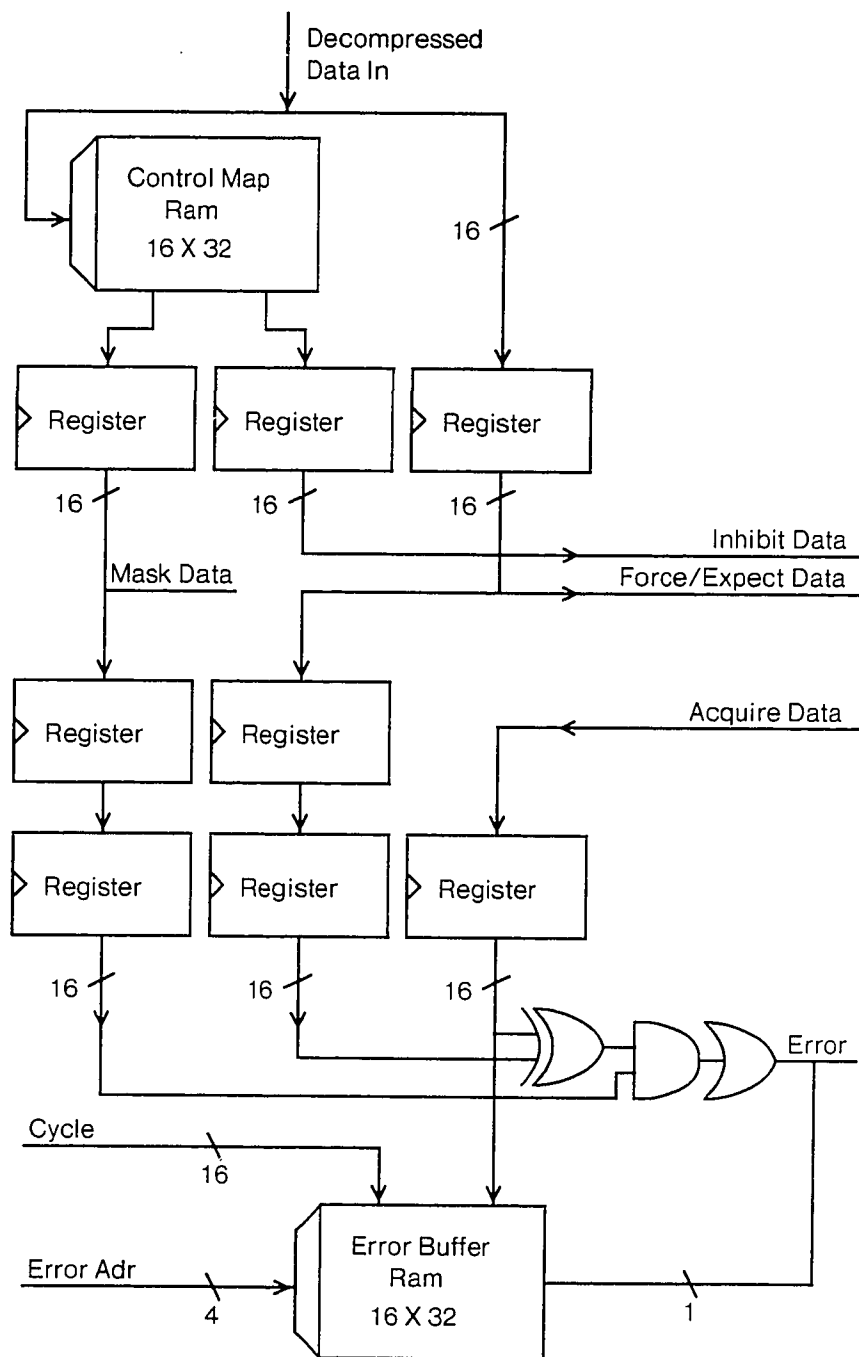


Figure 6.8: Error buffer

6.3.1 Pinout

The interface to the Testarossa chip as seen by the host processor is quite simple, consisting of a 10-bit multiplexed address/data bus plus a few clock and control signals, for a total of 22 pins. Including power, ground, and the DUT I/O connections, there are a total of 42 signals in the Testarossa pinout. These signals are described individually in the following list:

DUT[0..15], Input/Output - These are the sixteen signals which connect to the Device Under Test. Each can be programmed to have CMOS or TTL output and input levels, independent format, and timing.

IOAdrData[0..9], Input/Output - This is the general purpose port for accessing internal registers and RAMS in the Testarossa chip. It is a 10 bit bi-directional multiplexed address/data bus. The bus is completely asynchronous and can operate at any speed up to a few mega-Hertz.

IOAdr, Input - This signal is used to indicate that a valid internal register address is available on the IOAdrData lines. The address setup time before the trailing edge of IOAdr is 100 ns, and the hold time is 10 ns. The minimum IOAdr pulse width is 100 ns.

IORead, Input - This signal commands the chip to place the contents of the currently addressed internal register on the IOAdrData lines and turns on the IOAdrData output drivers. The data is valid 100 ns after the rising edge and remains valid as long as IORead is asserted.

IOWrite, Input - This signal commands the chip to load the currently addressed internal register with the data available on the IOAdrData lines. The write operation is flow-through, so the data must be valid 10 ns before IOWrite is

asserted and remain valid until 10 ns after IOWrite is de-asserted. The minimum IOWrite pulse width is 100 ns.

ChipSelect, Input - ChipSelect conditions the response of the chip to I/O commands (Read and Write). If ChipSelect is not asserted, then the device IOAddress lines are high impedance and no write operations will occur.

Start, Input - This signal must be synchronous to CycleClock and indicates that the chip should begin to execute the set of test vectors starting at address zero in the Vector RAM. Start is rising edge sensitive so it must first be de-asserted before re-running the same set of vectors. When the End Address is reached during vector execution, the tester will either return to the idle mode or will branch into a loop depending on the state of the Loop input.

Loop, Input - This signal must be synchronous to CycleClock and indicates that the chip should branch when the End Address is reached. While Loop is asserted, the vector execution will loop indefinitely. When Loop is de-asserted, execution will cease and the tester will return to the idle mode the next time that the End Address is encountered. Upon returning to idle, the last vector executed remains valid at the tester outputs.

Reset, Input - This signal is synchronous to CycleClock and indicates that the chip should return to the idle state, with no internal registers selected.

CycleClock, Clock, CycleClock/2, Input - CycleClock is the clock which determines the rate at which test vectors are delivered to the output. Clock and CycleClock/2 are used for delay generation as described in Chapter 4.

RefClock, Input - RefClock provides the reference timing edge for calibration purposes. It is important that the chip-to-chip skew of this signal be minimal.

VThreshold, Input - VThreshold provides the threshold voltage level used by the analog input comparators.

VddTTL, GndTTL, Input - These pins provide the high and low voltage rails for the TTL output pad driver. These voltages must be within the voltage range of Vdd and Gnd, nominally 2.0 volts and 0.8 volts respectively.

Vdd, Gnd, Input - These pins provide the high and low voltage rails for the CMOS output pad driver as well as the power supply voltages for the internal chip logic, nominally 5 and 0 volts respectively.

6.3.2 Register model

There are over 2700 bits of internal state contained in the control and status registers of Testarossa. These registers are divided into two main classes: the pin electronics registers and the decompressor control registers. The format of the registers for one channel of the pin electronics is shown in Figure 6.9. The bulk of the data in the registers is used for controlling the timing of the delay generators; this includes the Delay Shift Register (DSR), Inverter Chain (IC), and Edge Control (EC) units described in Chapter 4. The bits contained in these registers are un-encoded, that is, only one bit in the register is on (asserted high) at any time, and the MSB (bit 0) within a field always represents the minimum delay. For example, to select the third tap of the Inverter Chain delay element, bit 2 of IC0 would be set high, and all of the remaining bits in IC1 and IC2 would be low. The pin electronics registers are all eight bit quantities. These eight bits are right justified on the 10 bit IOAdrData bus of the Testarossa chip.

To compute the address of one of the control registers within a particular timing generator and channel, the three delay generators are assigned a numerical value: Sample = 0, Width = 1, Delay = 2. The base address for a particular delay generator

| Register | Dly. Gen. Offset | | Description |
|----------|------------------|-----|------------------|
| DSR0 | 0 | R/W | Delay Shift Reg |
| DSR1 | 1 | R/W | Delay Shift Reg |
| IC0 | 2 | R/W | Inverter Chain |
| IC1 | 3 | R/W | Inverter Chain |
| IC2 | 4 | R/W | Inverter Chain |
| ECR | 5 | R/W | Rising Edge Ctl |
| ECF | 6 | R/W | Falling Edge Ctl |

| Register | Chan. Offset | | Description |
|----------|--------------|-----|-------------|
| IOCtl | 7 | R/W | I/O Ctl |
| Format | F | R/W | Format |

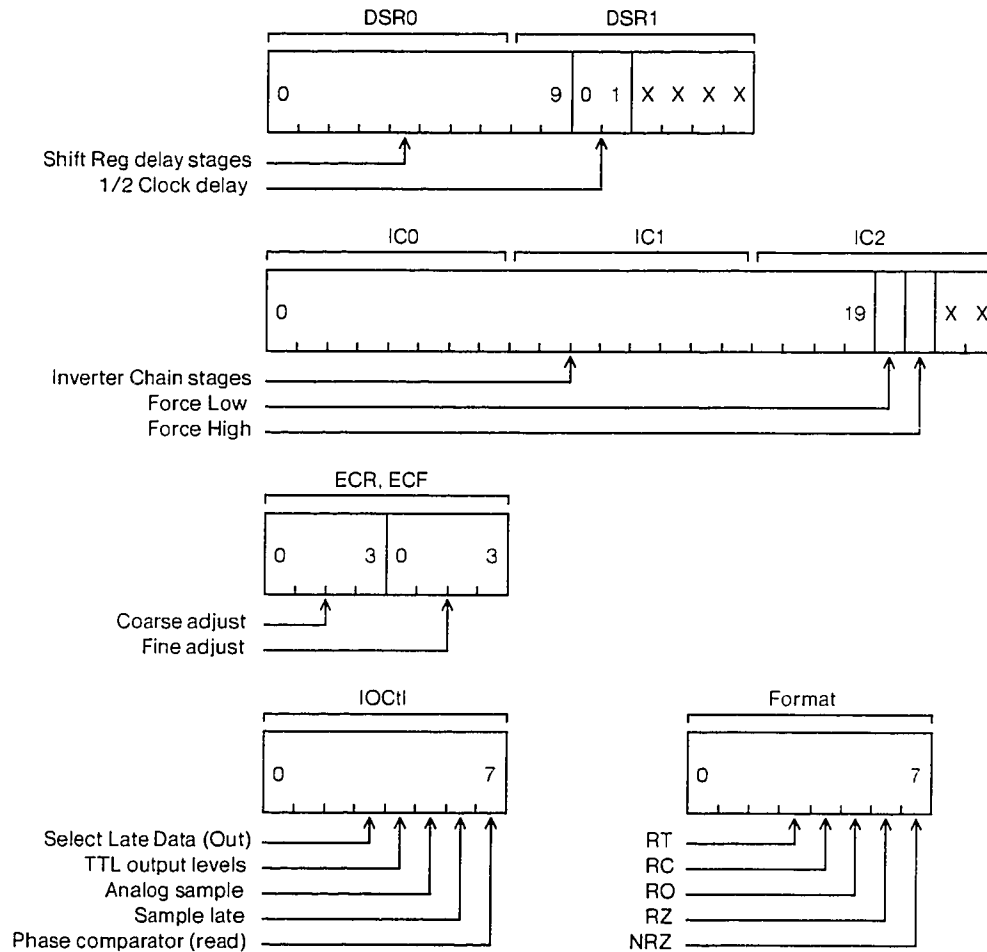


Figure 6.9: Pin electronics registers

within a channel is then given by :

$$\text{base address} = (\text{channel} * 24) + (\text{delay generator} * 8).$$

A control register for any of the delay generators is accessed by adding the offset value given in the table of Figure 6.9 to the base address. For the case of the IOCtl and Format registers, there is only one copy each per channel and the base address is simply the channel offset:

$$\text{base address} = (\text{channel} * 24).$$

The format of the decompressor control registers is shown in Figure 6.10. The addresses of these registers are absolute, so no offset calculation is necessary. All of the data available through the readable registers, with the exception of the RAM Read Data Register, is for debugging purposes and is not really of concern. The writeable registers contain the Loop and End addresses, as described in Chapter 5, and the interface to access the internal RAMS. All of the on-chip RAMS have a 40-bit data interface. To access a word in a RAM, the desired address is first written into the ExtRAdd register. If the operation is a write, then the four write data "bytes" are loaded into WD0 - WD3. Then, the appropriate control bit is set and cleared in the ExtRCtl register. In the event of a read operation, the data is then available in the four Read Data registers.

6.3.3 Programmer's interface

Given the complexity of the Testarossa chip, the programmers' interface for the device could be quite complex. In fact, with only a thin veneer of software, the interface can be reduced to a few key procedures. The following list represents the operations necessary to control all of the major functions of the device. The total length of the source code contained in the implementation of these procedures is about two thousand lines, including the calibration and compression utilities.

| Register | Add (hex) | | Description |
|----------|-----------|---|-------------------|
| Loop | 180 | W | Loop Add. |
| End | 181 | W | End Add. |
| ExtRAdd | 182 | W | External Ram Add. |
| ExtRCtl | 183 | W | External Ram Ctl. |
| RWD | 184-187 | W | Ram Write Data |
| DCtl | 188 | W | DRam Timing Ctl. |

| Register | Add (hex) | | Description |
|----------|-----------|---|------------------|
| Debug | 180 | R | Internal status |
| VAdd | 181 | R | Vector Ram Add. |
| HAdd | 182 | R | History Ram Add. |
| Cmd | 183 | R | Decompressor Cmd |
| RRD | 184-187 | R | Ram Read Data |

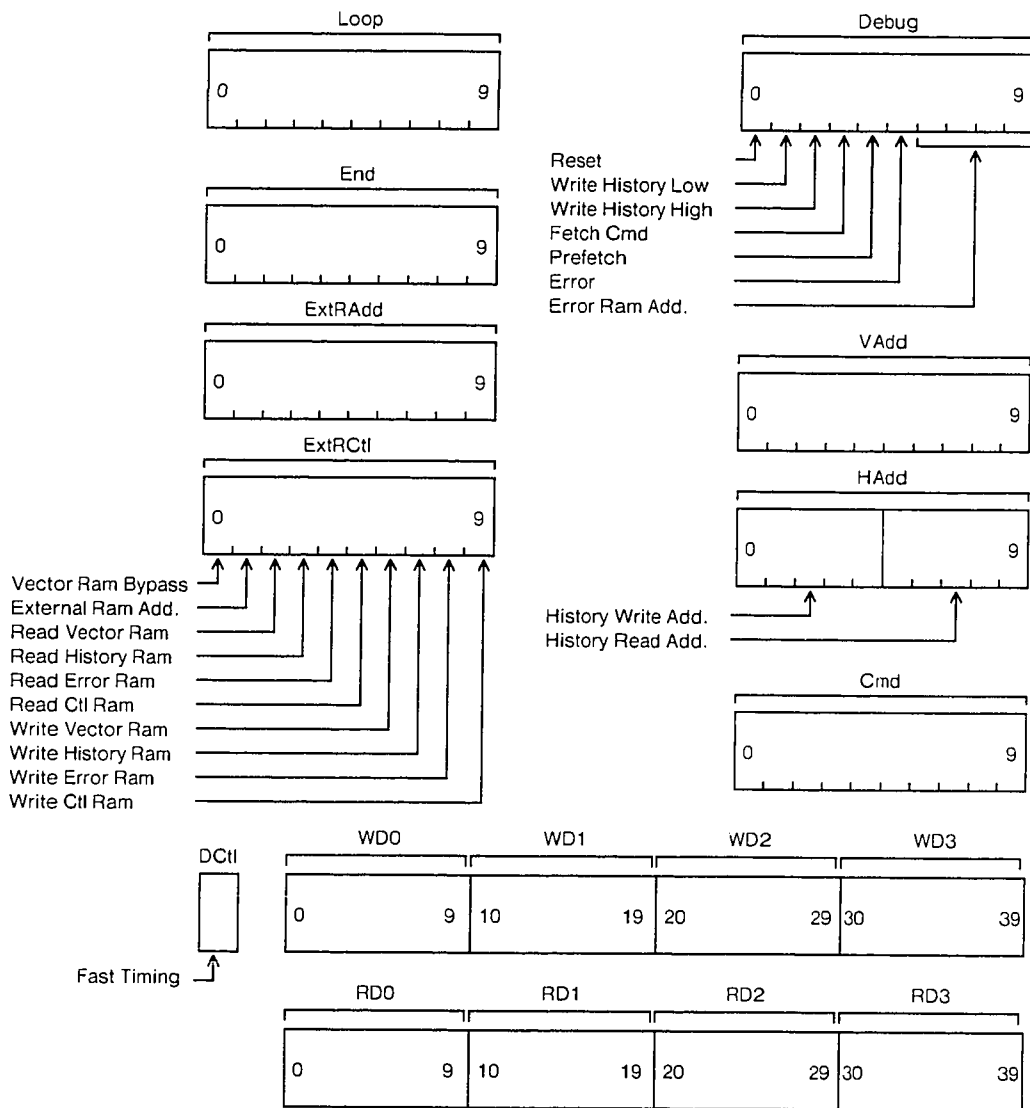


Figure 6.10: Decompressor control registers

InitSystem: PROC [];

Performs basic system initialization, such as resetting all chips, setting DUT outputs to high impedance, and zeroing DUT power supplies to avoid insertion damage.

LoadVectors: PROC [s: STREAM];

Takes a stream of vectors consisting of mask, inhibit, and force/expect data for each bit. Computes the contents of the Control Map and the uncompressed test vectors. Applies the compression operation to test vector data and loads Control Map memory and Vector memory.

SetTiming: PROC [ch: Channel, f: Format, dly, wid, samp: Ns];

Configures format, timing, and sample functions for a single channel in the system. Performs the calibration operation on the three delay generators of the channel.

SetAnalogLevels: PROC [ch: Channel, out, in: BOOLEAN];

Selects whether a channel's output and input levels are variable or CMOS levels.

SetClock: PROC [p: Period];

Sets the tester cycle period. Computes the correct ratio for Clock and Cycle-Clock and sets the clock divisor.

RunCtl: PROC [start, loop, reset: BOOLEAN];

Synchronously changes the state of the Start, Loop, and Reset pins of all chips in the system. Ensures that all chips act in lockstep.

GetErrors: PROC RETURNS [LIST OF [c: Cycle, d: Data]];

Used after a test is run to retrieve the contents of the Error Buffer. Returns a list of records containing the failing cycle numbers and the corresponding data.

6.4 Technology Constraints

A dominating factor in any custom IC design effort is the target technology in which the device will be fabricated. For Testarossa, the most aggressive technologies available through Mosis [To88] were 2μ and 1.6μ CMOS. The 1.6μ technology is reticle based and hence somewhat expensive, so the 2μ process was used for the initial prototype. Since the two technologies are design rule compatible, the device was designed with a rather large die size using the 2μ process, with the expectation that a later optical shrink could be performed on the device to gain the improved density, yield, and speed properties of the 1.6μ process after the design had been proven. The only impact of this approach on the 2μ die was a slightly oversized bonding pad. The maximum die size constraints were imposed by the available package cavity size and the tolerable yield. A source was located for a pin-grid array package with a 13.2×13.2 mm cavity which allowed a maximum die size of around 12×12 mm, leaving some space for a scribe line and die attach wash. Previous experience with the same vendor for the 2μ process indicated that a die of this area would have a yield in the range of 10 to 20%, which was considered acceptable.

6.5 Assembly

The Testarossa design contains over 200K transistors, 120K of these contained in RAM structures, and the remaining 80K distributed among the pin electronics, data path, and control circuitry. It would ordinarily be a formidable task for one person (albeit a graduate student) to attempt to assemble this many devices into a finished chip. Fortunately, a number of excellent assembly tools were available that greatly shortened the design cycle. These tools were provided by the Design and Architecture group in the Computer Science Laboratory at the Xerox Palo Alto Research Center [Ba88].

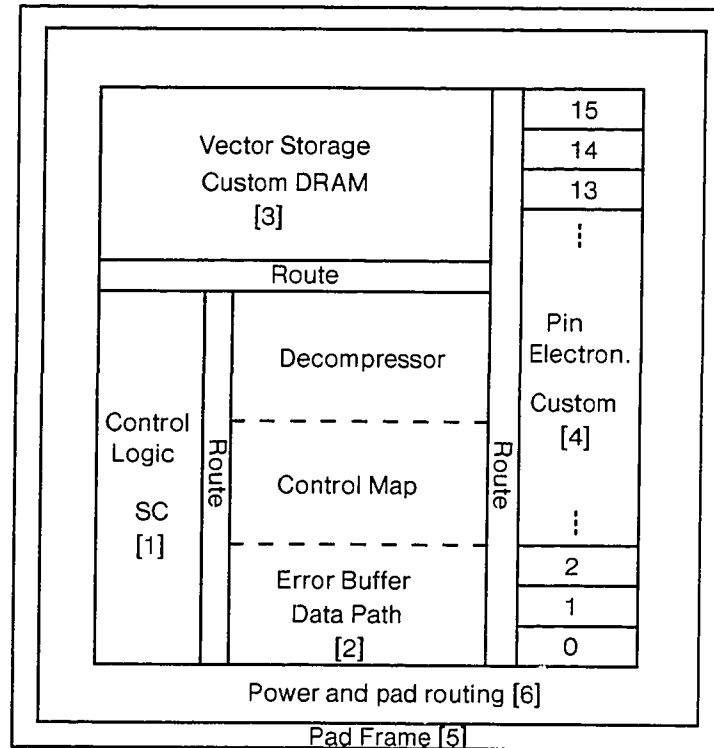


Figure 6.11: Testarossa floor plan

The tools consist of a number of different automatic layout generators, routers, and checkers, all driven from a schematic-based entry system.

Figure 6.11 shows a floorplan of the Testarossa chip depicting the various pieces of the assembly, the layout generator used for each piece, and the order in which the modules were assembled. Figure 6.12 is a plot of the completed chip. Construction begins in the lower left corner of the core of the chip with the control logic. This block contains mostly random logic for controlling the decompressor data path, interfacing to the external bus, and providing timing for the dynamic RAM. It is implemented with standard cell gates to minimize manual layout.

Figure 6.13 and Figure 6.14 illustrate two typical styles used for expressing the

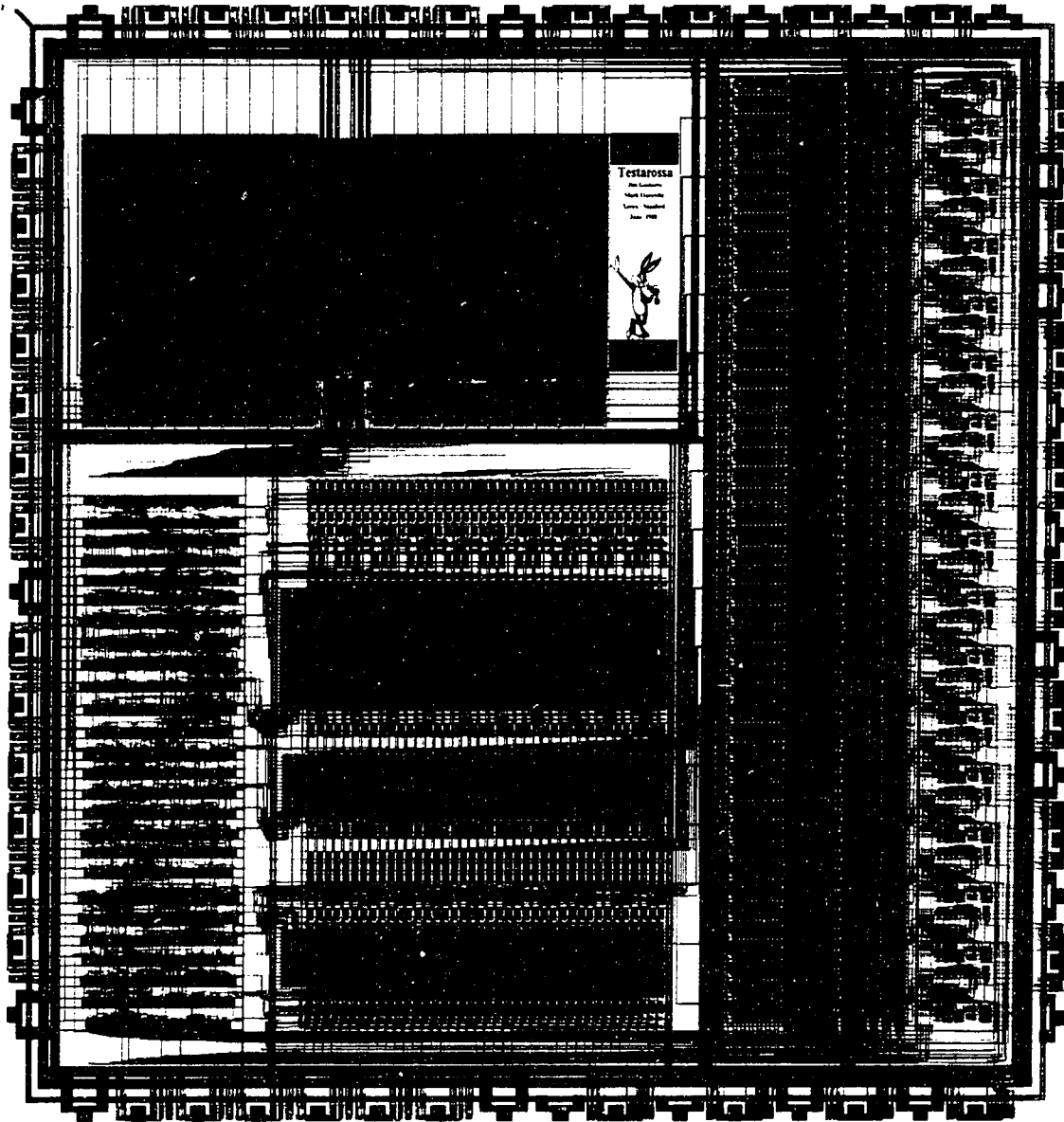


Figure 6.12: Testarossa layout

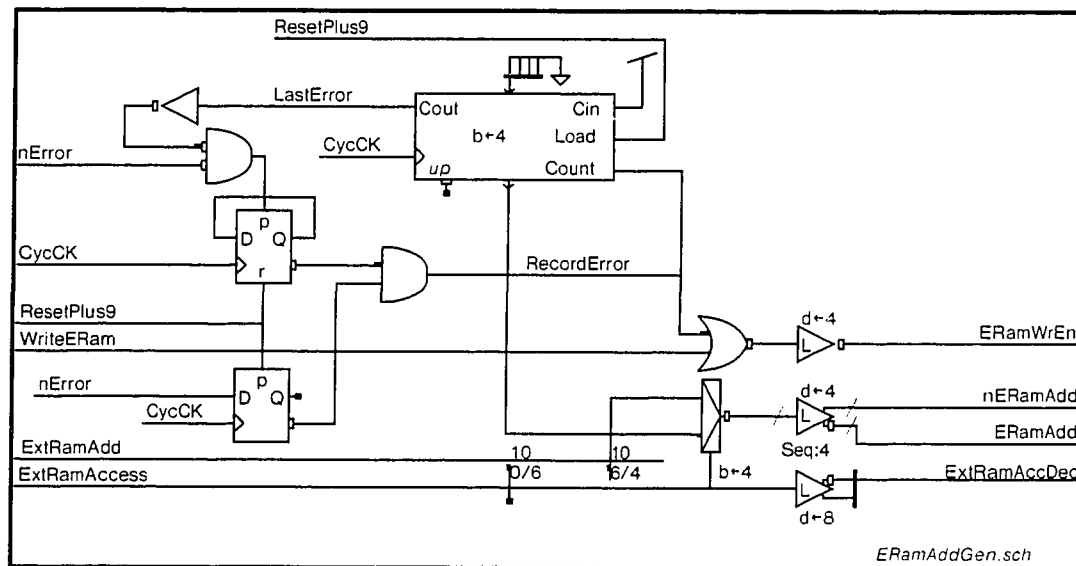


Figure 6.13: Typical schematic entry for standard cell control logic

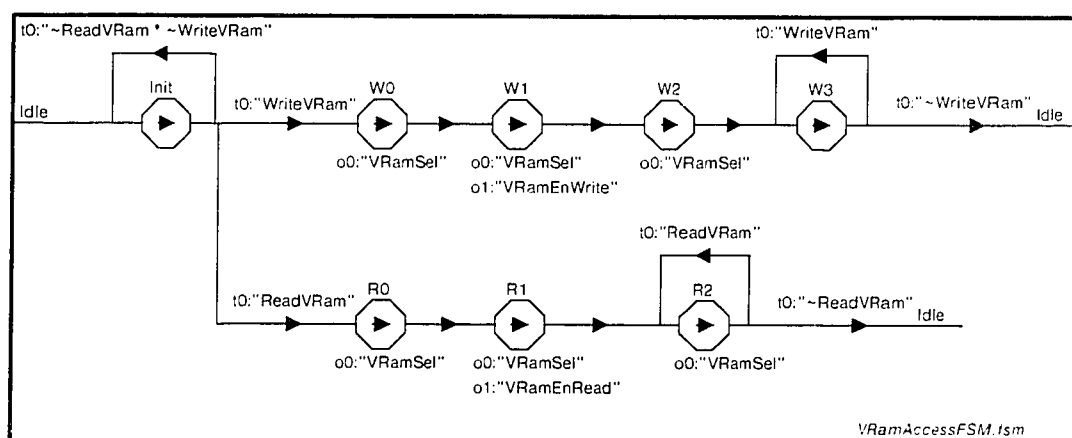


Figure 6.14: Graphical FSM description implemented in standard cell logic

control circuitry. Figure 6.13 shows a parameterized schematic that is part of the Error Buffer control logic. The schematic uses standard logic components, with the addition of the assignment texts on the logic icons, which allow easy specification of component parameters. For example, " $b \leftarrow 4$ " on the counter icon in the schematic indicates that a four bit counter should be generated, while " $d \leftarrow 4$ " on the inverter icon indicates the selection of a driver of four times the normal inverter strength.

Figure 6.14 illustrates another means of specifying logic. The picture is a graphical representation of a Mealy-type finite state machine used for arbitrating the DRAM refresh. The octagons are the state nodes, with the state names listed above and the asserted outputs listed beneath. The lines and arrows indicate the transitions, with conditional transitions having a logical expression above the arrow. This representation is automatically interpreted at the time the circuit is extracted to produce the standard cell gates necessary to implement the desired sequential function. These high level means of expressing control allow rapid development of complex logic.

The second step in the assembly process is the construction of the data path logic. This block contains the embedded RAMS for the History Buffer, Control Map, and Error Buffer, as well as the multiplexors, buffers, and pipeline registers needed in the decompressor data path. Figure 6.15 shows a sample schematic input used to generate the portion of the data path containing the History Buffer. All of the logic icons in the schematic have manually drawn layout associated with them with the exception of the multiplexors, which are automatically generated. The compiler sequences the base cells by the width of the data path and then automatically generates the horizontal and vertical routing to connect them together. Each of the four columns of icons in the figure represents a separate 10 bit data path; these are interleaved to reduce the horizontal routing cost. The top three rows of cells in the example implement the History Buffer RAM. The History Buffer is a 64×20 RAM but, as shown in the figure, it is actually implemented as a 32×40 RAM to allow the data path area to

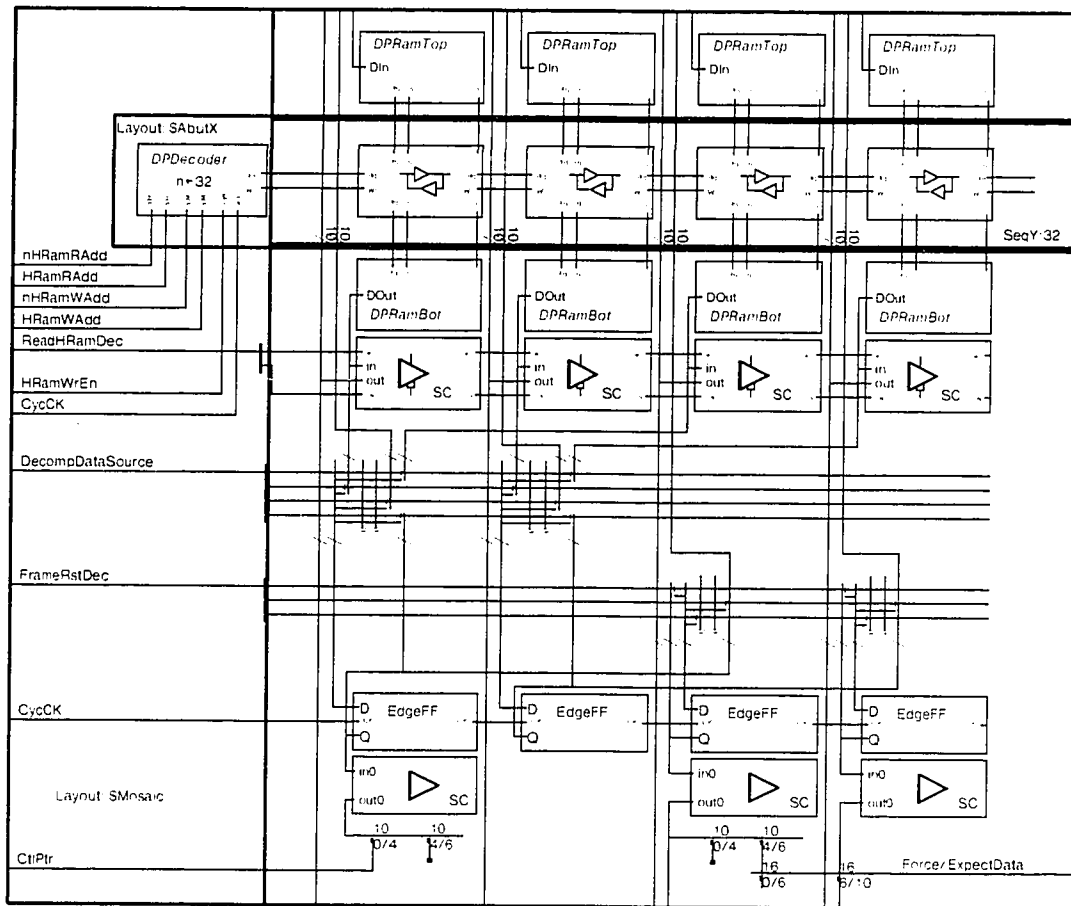


Figure 6.15: Decompressor data path segment

be used efficiently. The RAM array is generated by creating a 40 bit wide row of RAM cells using the data path compiler, sequencing this 32 times in Y to form the array, then abutting a program-generated decoder on the left. The result is then sandwiched between two more data path segments that contain the bit line drivers on top and the read logic on the bottom. The remainder of the data path logic is generated in a similar manner and finally automatically routed to the standard cell control logic on the left.

The next step of the construction process is the addition of the full custom dynamic RAM. Another routing channel is used to connect the address, data, and control lines of the RAM to the standard cell and data path elements. Next, a single channel of the full custom pin electronics is sequenced 16 times in Y and abutted to the right side. One more routing channel connects the Force Data and Inhibit Data busses from the data path to the pin electronics. Finally, the pad frame is generated, and the global power and pad routing is performed.

At each step of the assembly process, the netlist of the generated layout is compared to the schematic netlist to ensure that cell abutment produces the desired connectivity. This comparison also acts as a check for the automatic layout generators, such as the routers and multiplexor generators. The compilation of the entire chip with netlist checking takes approximately 20 MIPS-hours of CPU time. However, small changes can be made to sub-modules with only incremental compilation, allowing fast turnaround for small changes.

A number of features have been incorporated into the layout to enhance the testability of the device. All of the internal RAMS are available for reading and writing from the external bus, as are the over 2K register bits contained in the pin electronics. In the control section, a number of key control fields of the decompressor are readable, such as the RAM addresses, the current command, and the state of the prefetcher. When used together, these features enable each of the sub-modules in the device to

be tested independently. For example, the decompressor can be operated without a functioning vector DRAM by bypassing the RAM and supplying the vectors externally. The decompressed result can also be verified without use of the pin electronics by examining the state of the History Buffer from the external bus. Such features ensured that the maximum amount of data could be gleaned from the part in spite of processing or logical defects.

6.6 Results

The layout and simulation of Testarossa were completed in about six months and the device was fabricated through the MOSIS implementation service. The first silicon was nearly 100% functional, which is a testimony to the methodology of schematic entry, layout versus schematic verification, and simulation, simulation, and more simulation. The only defect in the chip was a timing error in the DRAM which caused some crosstalk between the data columns of the RAM. This prevented more than one column of the DRAM from being used. Fortunately, the column selects were derived from the high order address bits, so that the first 128 words in the RAM could be used reliably. This provided enough vector storage space to allow thorough testing of the decompressor.

The maximum speed of the device was just over 17 MHz, which was somewhat slower than the anticipated 25 MHz. The limiting factor was the poor access time of the DRAM, which was related to the crosstalk problem mentioned above. When the decompressor was operated in a debugging mode that bypassed the DRAM and executed a four instruction loop out of the RAM Write Data register, a maximum speed of 33 MHz was attained. This is indicative of what could be achieved with a better RAM technology.

The DUT waveform exhibited good edge and timing resolution characteristics.

The rising edge kink effect observed in the experimental pin electronics chip was eliminated in this device yielding an unloaded rising edge speed of 3 ns. The edge resolution was 600 ps, the same as that of the earlier device. Of 27 parts tested, 4 were fully functional for a 14% yield. The temperature sensitivity of the inverter chain delay elements was measured at 20 pS per degree centigrade per stage and the voltage sensitivity was 70 pS per volt per stage, thus providing adequate delay stability. Calibration routines were written for the device and were successful in calibrating all edges to better than 1 ns over a wide range of delays and cycle frequencies.

After the design was verified using the 2μ process, the follow-on 1.6μ device was also fabricated with only one change to the DRAM control circuitry to correct the timing error. This device was 100% functional and yielded better than 40% good die. The technology shrink improved the maximum operating speed, resulting in 25 MHz operation and improved the edge resolution to 500 ps.

Chapter 7

Conclusions

7.1 Summary

This thesis has presented a new architecture for building high-performance single-chip VLSI testers. New circuit and system designs for the pin electronics have been shown which solve the problems of achieving high timing accuracy for output edge placement and input sample capture in a relatively slow base technology such as CMOS. These techniques provide sufficient density advantages over traditional methods that they enable the implementation of a multi-channel device with true tester-per-pin characteristics. A reduction in the size of the data generator portion of the tester has also been achieved through the use of data compression. A number of compression methods have been examined to determine their applicability to reducing the memory requirements of test vectors. One algorithm was identified which averaged good compression characteristics with only modest decompression hardware requirements. Finally, all of these ideas have been brought together in the design and successful implementation of Testarossa: a single-chip 16-channel tester containing all of the functions of pin electronics, vector storage, decompression, and acquisition logic.

7.2 Future Work

There are many ways in which the architecture can be improved and extended as new technologies become available. The current 2μ Testarossa implementation is rather behind the state of the art in terms of process technology. A 0.8μ process would provide a four-fold increase in effective die area while still allowing a 20% shrink in the actual die dimensions. One way to use the additional die area would be to simply replicate the existing logic to provide more DUT channels per chip, allowing an even denser system to be built. However, this does not seem to be the most advantageous use of the added silicon area. A better approach would be to use the space to increase the on-chip resources for improving system capabilities. For example, the current limit of 16 Control Map entries is somewhat restrictive. Increasing it to 32 or 64 entries would make the existence of the map more transparent.

Another area that could be improved is the timing and format registers. Currently these registers are statically loaded during the calibration process prior to running the test, making the wave shape and edge placement fixed for the duration of the test. An improved version would have a number of these registers available, allowing dynamic changes in format and timing. This would provide greater flexibility in the timing properties of the applied vectors. The selection of the timing characteristics to be applied to a vector could be done through the Control Map. Currently only 32 bits of the 40-bit map entries are used. Some of these remaining 8 bits could be used to select the desired timing entry, allowing timing change on a per-vector basis with no increase in vector size.

A third area that could benefit from additional die area is the Vector Storage RAM. This is the most likely candidate for consuming the majority of the added area, since the 10K-vector storage capacity of the current implementation is rather limited, compared to the majority of testers available. In Testarossa, the DRAM occupies only

14% of the total die area. If all of the additional space were devoted to vector storage, the storage capacity would increase by more than a factor of 20, providing over 200K vectors on-chip (compressed average). As the percentage of die area devoted to vector storage increases, the benefit of the vector decompressor becomes more evident. The ratio of vector DRAM to decompressor areas in Testarossa is approximately 1:1. Given that the average compression factor is about 5, the effective compression ratio is only 2.5:1, since the DRAM area could be doubled if the decompressor were not present. The effect of a process shrink is to increase the DRAM size while decreasing the decompressor area making the effective compression ratio much closer to the average. Even greater storage capacity could be achieved by employing a technology which provided a real DRAM storage cell, such as the trench capacitor process, as opposed to the poly-diffusion cell capacitor used in Testarossa. This would improve the bit-density by another factor of eight to ten, pushing the vector storage capacity well into the mega-vector regime. Table 7.1 shows a comparison of static and dynamic RAM implementations contrasting the generic 2μ digital technology used for Testarossa with the best RAM technologies currently available [Lu88, Ma87, Ok88].

| Technology | Type | Cell | Size (μ^2 , est.) | Cycle (ns, est.) |
|------------|---------|----------|------------------------|------------------|
| 2μ | Static | 6T | 1600 | 15 |
| 2μ | Dynamic | Poly-Dif | 300 | 35 |
| 0.8μ | Static | 4T | 70 | 8 |
| 0.8μ | Dynamic | Trench | 15 | 30 |

Table 7.1: Comparison of static and dynamic RAM implementations

An alternative approach for the vector storage is to use static RAM instead of DRAM. The major tradeoff between these two choices is storage density versus access speed. The data path of the tester and decompressor is a fairly simple pipelined architecture, so the limiting factor in the tester performance is the RAM cycle time. By going to a fast static RAM process, the performance of the system could be dramatically improved, but at the expense of reduced vector storage capacity.

Advanced process technology can be applied to other aspects of the architecture as well. Bipolar transistors, such as those available in a Bi-CMOS process, would help in several areas of the design. For example, the higher gain of the bipolar devices would allow the current drive of the output pad driver transistor to be improved while at the same time reducing the capacitive load of the driver on the input. This would enable faster output rise-times and lower DUT output loading. Bipolar transistors could also be used in the calibration phase comparator and input sample comparator. The higher gain-bandwidth product of the bipolar devices would result in greater timing accuracy.

In addition to architectural improvements to the Testarossa device, some work remains to be completed before a complete multi-hundred pin tester system can be assembled. The bulk of the work is in the design of the reference clock generator circuit. The requirements of this circuit are that it produces a square waveform at the tester cycle frequency with a phase relationship to the system clock that is programmable with sub-nanosecond resolution and accuracy over the range of a microsecond or so. It is no surprise that these are the same requirements as those of the tens or hundreds of individual clock generator circuits used in commercial testers. It should be relatively easy, then, to leverage off of these earlier designs to implement a high quality generator using a semi-custom ECL device. Alternatively, there are CMOS approaches employing phase-locked loop principles that could be employed that would keep the design of the entire system in one technology [JB87].

Bibliography

- [Ab84] W. Abbott IV,
Timing Specification Conformance of VLSI Test Systems,
Proceedings of the 1984 International Test Conference, pp. 105-112, October 1984.
- [Al83] R. Albrow,
Test Pattern Compaction in VLSI Testers,
Proceedings of the 1983 International Test Conference, pp. 12-17, October 1983.
- [Ba83] M. Barber,
Subnanosecond Timing Measurements on MOS Devices Using Modern VLSI Test Systems,
Proceedings of the 1983 International Test Conference, pp. 170-180, October 1983.
- [Ba84] M. Barber,
Fundamental Timing Problems in Testing MOS VLSI on Modern ATE,
IEEE Design and Test, pp. 482-489, August 1984.

- [Ba88] R. Barth, L. Monier, B. Serlet, and P. Sindhu,
VLSI Design Aids: Capture, Integration, and Layout Generation,
Xerox PARC Technical Report, CSL-88-1, July 1988.
- [Bi83] S. Bisset,
The Development of a Tester-Per-Pin VLSI Test System Architecture,
Proceedings of the 1983 International Test Conference, pp. 151-155, October 1983.
- [Bo81] P. Bottorff,
Functional Testing Folklore and Fact,
Digest of Papers of the 1981 International Test Conference, pp. 463-464, October 1981.
- [BP85] G. Bowers and B. Pratt,
Low Cost Testers: Are They Really Low Cost?,
IEEE Design and Test, pp. 20-28, June 1985.
- [CC87] S. Cohen and J. Chen,
Maintaining Timing Accuracy in High Pin-Count VLSI Module Test Systems,
Proceedings of the 1987 International Test Conference, pp. 779-789, September 1987.
- [CF83] M. Catalano, R. Feldman, R. Krutiansky, and R. Swan,
Individual Signal Path Calibration for Maximum Timing Accuracy in a High Pincount VLSI Test System,
Proceedings of the 1983 International Test Conference, pp. 188-192, October 1983.

- [CR81] P. Chang, E. Richards, and D. Richter,
The PIN Module: A High Accuracy Concept in Very High Frequency Pin Electronics,
Digest of Papers of the 1981 International Test Conference, pp. 154-166, October 1981.
- [Da87] M. Dahl,
Closed-Loop Error Correction: A Unique Approach to Test System Calibration,
Proceedings of the 1987 International Test Conference, pp. 772-778, October 1987.
- [De83] L. Deerr,
Automatic Calibration for a VLSI Test System,
Proceedings of the 1983 International Test Conference, pp. 181-187, October 1983.
- [EW77] E. Eichelberger and T. Williams,
A Logic Design Structure for LSI Testing,
Proceedings of the 14th Design Automation Conference, New Orleans, pp. 462-468, June 1977.
- [Fe78] M. Ferland,
Device Output Loading,
Digest of Papers of the 1978 Semiconductor Test Conference, pp. 130-132, October 1978.
- [FG88] E. Fiala and D. Greene,
Data Compression with Finite Windows,
to appear in CACM, spring 1989.

- [Ga84] R. Garcia,
The Fairchild Sentry 50 Tester: Establishing New ATE Performance Limits,
IEEE Design and Test, pp. 101-109, May 1984.
- [He78] J. Healy,
An Analysis of Trends in Complex LSI Testing Strategies,
Digest of Papers of the 1978 Semiconductor Test Conference, pp. 59-64, October 1978.
- [He83] R. Herlein,
Optimizing the Timing Architecture of a Digital LSI Test System,
Proceedings of the 1983 International Test Conference, pp. 200-208, October 1983.
- [Hu52] D. Huffman,
A Method for the Construction of Minimum-Redundancy Codes,
Proceedings of the I. R. E., Vol. 40, pp. 1098-1101, 1952.
- [HU85] J. Healy and G. Ure,
A Method of Reducing ATE System Error Components and Guaranteeing Subnanosecond Measurement Accuracies,
Proceedings of the 1985 International Test Conference, pp. 191-202, November 1985.
- [IM85] *Logic Master Series Operator's Manual,*
Integrated Measurement System Inc., Part no. 900-0001-P01,
March 1985.
- [JB87] D. Jeong, G. Borriello, D. Hodges, and R. Katz,
Design of PLL-based Clock Generation Circuits,

- IEEE Journal of Solid-State Circuits, Vol. SC-22, pp. 255-261,
April 1987.
- [JM84] P. Jackson, G. de Mare, and A. Esser,
Compaction Technique Universal Pin Electronics,
Proceedings of the 1984 International Test Conference, pp. 471-
480, October 1984.
- [Ka85] T. Kazamaki,
Milestones of New-Generation ATE?,
IEEE Design and Test, pp. 83-89, October 1985.
- [Ke86] M. Keating,
Fundamental Limits to Timing Accuracy,
Proceedings of the 1986 International Test Conference, pp. 756-
762, October 1986.
- [KM78] T. Kazamaki et al,
*Trial Model of 100 MHz test Station for High Speed LSI Test
System*,
Digest of Papers of the 1978 Semiconductor Test Conference, pp.
139-145, October 1978.
- [Kn68] D. Knuth,
The Art of Computer Programming,
Fundamental Algorithms, Vol. 1, pp. 402-404, 1968.
- [Kn85] D. Knuth,
Dynamic Huffman Coding,
Journal of Algorithms, Vol. 6, pp. 163-180, 1985.

- [Lu88] N. Lu, H. Chao, W. Hwang, W. Henkels, T. Rajeevakumar, H. Hanafi, L. Terman, and R. Franch,
A 20-ns 128K X 4 High-Speed DRAM with 330-Mbit/s Data Rate,
IEEE Journal of Solid-State Circuits, Vol. SC-23, pp. 1140-1149,
October 1988.
- [Ma87] K. Mashiko, M. Nagatomo, K. Arimoto, Y. Matsuda, K. Furu-
tani, T. Matsukawa, M. Yamada, T. Yoshihara, and T. Nakano,
*A 4-Mbit DRAM with Folded-Bit-Line Adaptive Sidewall-Isolated
Capacitor (FASIC) Cell*,
IEEE Journal of Solid-State Circuits, Vol. SC-22, pp. 643-650,
October 1987.
- [Me84] J. Meindl,
Ultra-Large Scale Integration,
IEEE Transactions on Electron Devices, Vol. ED-31, no. 11, pp.
1555-1561, November 1984.
- [MH87] J. Miyamoto and M. Horowitz,
A Single-Chip High-Speed Functional Tester,
IEEE Journal of Solid-State Circuits, Vol. SC-22, pp. 820-828,
October 1987.
- [MS79] H. Maruyama, S. Sugamori, T. Sudo, and Y. Ichimiya,
A 100 MHz Test Station for High Speed LSI Testing,
Digest of Papers of the 1979 Test Conference, pp. 369-376, Oc-
tober 1979.
- [MS87] R. Muething and C. Saikley,
Integrated Pin Electronics: A Path Toward Affordable Testing of

- High-Pin Count ASIC Devices*,
Proceedings of the 1987 International Test Conference, pp. 883-887, September 1987.
- [NH84] Y. Nishimura, M. Hamada, and Y. Hayasaka,
A New Timing Calibration Method for High Speed Memory Test,
Proceedings of the 1984 International Test Conference, pp. 113-117, October 1984.
- [Ok88] H. Okuyama, T. Nakano, S. Nishida, E. Aono, H. Satoh, and S. Arita,
A 7.5-ns 32K X 8 CMOS SRAM,
IEEE Journal of Solid-State Circuits, Vol. SC-23, pp. 1054-1059, October 1988.
- [Pa76] R. Pasco,
Source Coding Algorithms for Fast Data Compression,
Ph.D Dissertation, Stanford University, 1976.
- [Pe85] D. Petrich,
Achieving Accurate Timing Measurements on TTL/CMOS Devices,
IEEE Design and Test, pp. 33-41, August 1986.
- [Pi78] R. Piety,
Low Cost Test System for Integrated Circuit Development,
Digest of Papers of the 1978 Semiconductor Test Conference, pp. 133-138, October 1978.
- [Sk80] K. Skala,
Continual Autocalibration for High Timing Accuracy,

- Digest of Papers of the 1980 Test Conference, pp. 111-116, November 1980.
- [SM80] R. Sherman and D. Madsen,
Low Cost Pattern Generator for Testing Digital LSI Devices,
Digest of Papers of the 1980 Test Conference, pp. 68-73, November 1980.
- [SM87] C. Saikley and R. Muething,
A Rapid, Low-cost Technique for Precise AC Calibration in a Focused ASIC Tester,
Proceedings of the 1987 International Test Conference, pp. 766-771, October 1987.
- [SY81] S. Sugamori, K. Yoshida, H. Maruyama, S. Kamata, and R. Sudo,
Analysis and Definition of Overall Timing Accuracy in VLSI Test System,
Digest of Papers of the 1981 International Test Conference, pp. 143-153, October 1981.
- [Te83] C. Terman,
Simulation Tools For Digital LSI Design,
Massachusetts Institute of Technology, Cambridge, Tech. Rep. MIT-LCS-TR-304, September 1983.
- [To88] C. Tomovich,
Mosis - A Gateway to Silicon,
IEEE Circuits and Devices, pp. 22-23, March 1988.
- [We83] B. West,
Attainable Accuracy of Autocalibrating VLSI Test Systems,

Proceedings of the 1983 International Test Conference, pp. 193-199, October 1983.

[Xe86]

Xerox Corporation,
Interpress Electronic Printing Standard,
Version 3.0, XNSS 048601, Stamford, CT, January 1986.

[ZL77]

J. Ziv and A. Lempel,
A Universal Algorithm for Sequential Data Compression,
IEEE Transactions on Information Theory, Vol. IT-23, No. 3,
May 1977.

89

2

5

8

6

9

LOW
CONTRAST
PHOTOS

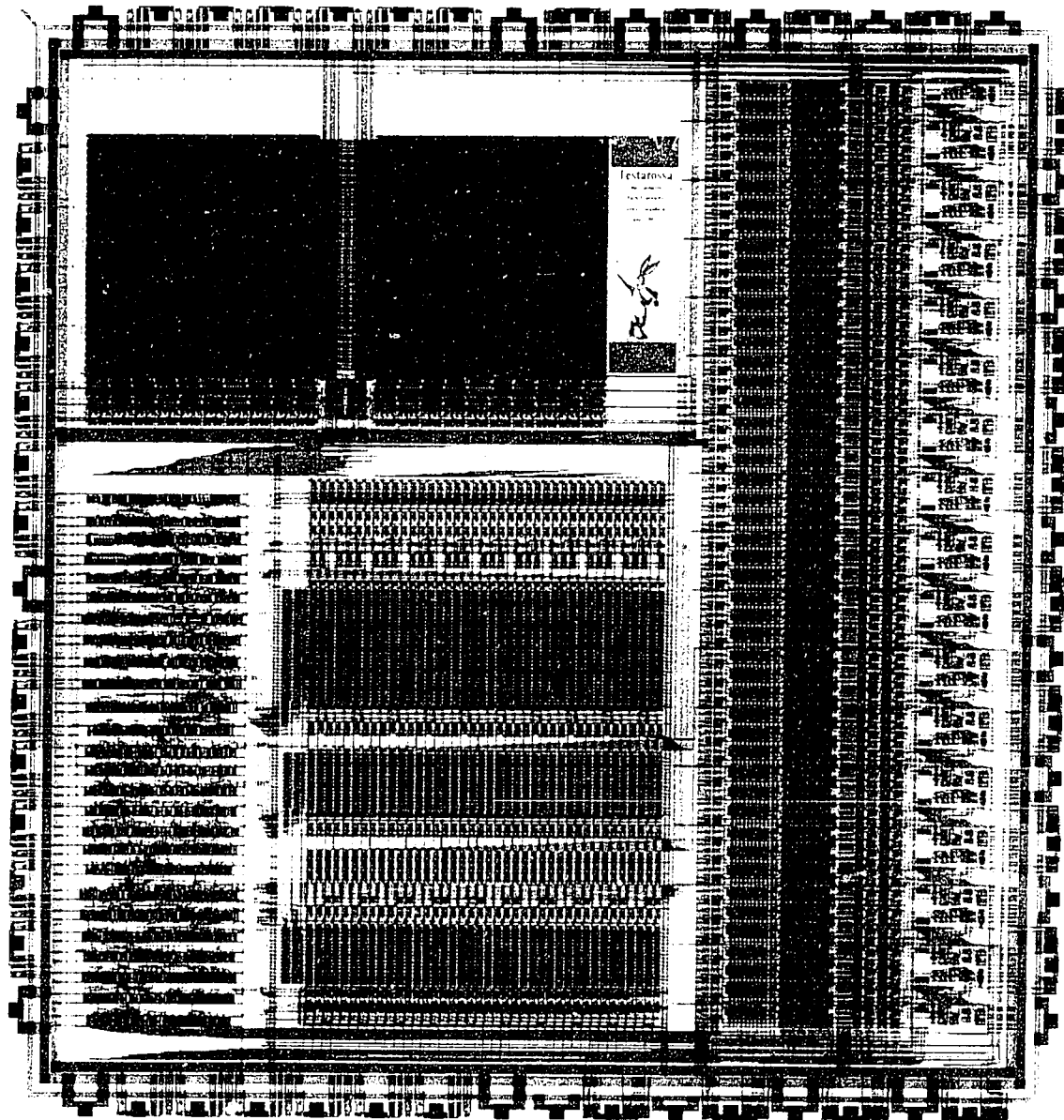


Figure 6.12: Testarossa layout