DOT/FAA/AR-95/31

Office of Aviation Research
Washington, D.C. 20591

Design, Test, and Certification Issues for Complex Integrated Circuits

August 1996

Final Report

U.S. Department of Transportation
Federal Aviation Administration

Technical Report Documentation Page

1. Report No. DOT/FAA/AR-95/31	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle DESIGN, TEST, AND CERTIFICATION ISSUES FOR COMPLEX INTEGRATED CIRCUITS		5. Report Date August 1996
		6. Performing Organization Code
7. Author(s) L. Harrison and B. Landell		8. Performing Organization Report No.
9. Performing Organization Name and Address Galaxy Scientific Corporation 2500 English Creek Avenue Building 11 Egg Harbor Township, NJ 08234-5562		10. Work Unit No. (TRAIS)
		11. Contract or Grant No. DTFA03-89-C-00043
12. Sponsoring Agency Name and Address U.S. Department of Transportation Office of Aviation Research Washington, D.C. 20591		13. Type of Report and Period Covered Final Report
		14. Sponsoring Agency Code AAR-421

15. Supplementary Notes

Peter J. Saraceni, William J. Hughes Technical Center Program Manager, (609) 485-5577, Fax x-4005, saracenp@admin.tc.faa.gov

16. Abstract

This report provides an overview of complex integrated circuit technology, focusing particularly upon application specific integrated circuits. This report is intended to assist FAA certification engineers in making safety assessments of new technologies. It examines complex integrated circuit technology, focusing on three fields: design, test, and certification. It provides the reader with the background and a basic understanding of the fundamentals of these fields. Also included is material on the development environment, including languages and tools.

Application specific integrated circuits are widely used in Boeing 777 fly-by-wire aircraft. Safety issues abound for these integrated circuits when they are used in safety-critical applications. Since control laws are now executed in silicon and transmitted from one integrated circuit to another, reliability issues for these integrated circuits take on a new importance. This report identifies certification risks relating to the use of complex integrated circuits in fly-by-wire applications.

17. Key Words Application specific integrated circuit, Avionics, Certification, Design for test, Field programmable gate array, Fly-by-wire, Hardware description language, Reliability, Simulation, Software, State machine, Submicron design, Synthesis, Validation, Verification.	18. Distribution Statement This document is available to the U.S. public through the National Technical Information Service, Springfield, Virginia 22161.

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 191	22. Price

Form DOT F1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

θ_{CA}	Case-to-Ambient Thermal Resistance
θ_{JC}	Junction-to-Case Thermal Resistance
ac	Alternating Current
AC	Advisory Circular
ACO	Aircraft Certification Office
AHDL	Analog Hardware Description Language
ALU	Arithmetic Logic Unit
ARINC	Aeronautical Radio, Inc.
ASIC	Application Specific Integrated Circuit
ASM	Algorithmic State Machine
ATE	Automatic Test Equipment
ATPG	Automatic Test Pattern Generation
BIST	Built-In Self-Test
BSDL	Boundary Scan Description Language
BSR	Boundary Scan Register
BST	Boundary Scan Test
CAD	Computer Aided Design
CANCER	Computer Analysis of Nonlinear Circuits, Excluding Radiation
CDFG	Control-Data flow Graph
CE	Certification Engineer
CFI	CAD Framework Initiative
CMOS	Complimentary Metal-Oxide Semiconductor
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
CUT	Circuit Under Test
dc	Direct Current
DFG	Data Flow Graph
DFT	Design For Testability
DIP	Dual In-line Package
DMA	Direct Memory Access
DUT	Device Under Test
ECL	Emitter-Coupled Logic
EDA	Electronic Design Automation
EDAC	Error Detection and Correction
EDIF	Electronic Design Interchange Format
EIA	Electronic Industries Association
EPLD	Erasable Programmable Logic Device
EPROM	Erasable Programmable Read-Only Memory
ESD	Electrostatic Discharge
ESDA	Electronic System Design Automation
ESTA	Electronic System Test Automation
FAA	Federal Aviation Administration

FAR	Federal Aviation Regulation
FFB	Fast Function Block
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FSMD	Finite State Machine with Data Path
HCPLD	High Capacity Programmable Logic Device
HDL	Hardware Description Language
HDLC	High-Level Data Link Control
HF	High Frequency
IC	Integrated Circuit
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
I/O	Input/Output
I_{DD}	CMOS Device dc Power Supply Current
I_{DDQ}	Quiescent State of Power Supply Current I_{DD}
JTAG	Joint Test Action Group
LF	Low Frequency
LFSR	Linear Feedback Shift Register
LIR	Left Instruction Register
LRU	Line Replaceable Unit
LSI	Large Scale Integration
LSSD	Level Sensitive Scan Design
mA	Milliampere
MHz	Megahertz
MOS	Metal-Oxide Semiconductor
MSI	Medium-Scale Integration
MUX	Multiplexer
NAND	Inverting Logical AND Gate
NLFSR	Nonlinear Feedback Shift Register
NMOS	Negative-Well MOS
NOR	Inverting Logical OR Gate
ns	Nanosecond
ORA	Output Response Analyzer
P	Power
PAL	Programmable Array Logic
PC	Personal Computer
PDES	Product Data Exchange Specification
PLD	Programmable Logic Device
PMOS	Positive-Well MOS
PROM	Programmable Read-Only Memory
QTAG	Quality Test Action Group
RAM	Random Access Memory
RC	Resistance-Capacitance
RIR	Right Instruction Register

ROM	Read-Only Memory
RTCA	Requirements and Technical Concepts for Aviation (formerly Radio Technical Commission for Aeronautics)
RTL	Register Transfer Level
SAE	Engineering Society for Advancing Mobility Land Sea Air and Space (formerly Society of Automotive Engineers)
SC-180	Special Committee 180
SC	Scan Control
SCR	Silicon-Controlled Rectifier
SEU	Single Event Upset
SI	Scan In
SO	Scan Out
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random Access Memory
SRC	Scan Register Chain
SSI	Small-Scale Integration
T_A	Ambient Temperature
TAP	Test Access Port
T_C	Case (Package) Temperature
TC	Test Clock
TCK	Tester Clock
TDI	Test Data In
TDO	Test Data Out
T_J	Junction Temperature
TMS	Test Mode Select
TPG	Test Pattern Generator
TRST	Test Reset
TTM	Time-To-Market
UIM	Universal Interconnect Matrix
V&V	Verification and Validation
V	Volt
V_{cc}	Collector Supply Voltage
V_{DD}	CMOS Device dc Power Supply Voltage
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration

EXECUTIVE SUMMARY

If one recognizes that there is concern over software correctness issues for safety-critical systems, then attention is also warranted for complex hardware design of safety-critical systems. Digital hardware design has been on a rapid track toward massive integration at the system level, as well as at the integrated circuit level. Entire systems implemented on a single application specific integrated circuit (ASIC) will soon be commonplace.

Not only is design correctness a concern, but demonstrating that there are no silicon defects can be a major problem for developers of complex ASICs. Due to an ASIC's complexity, complete ASIC testing has become an impossible task. In addition, no single test technique can demonstrate defect-free silicon. Combinations of tests are required for uncovering different failure modes. Issues that pose threats to reliability include signal integrity, accurate timing parameters for simulation, and numerous other submicron technology-related factors.

Following are some of the significant conclusions from this report:

- Due to their complexity, ASICs can incorporate significant portions of a system's functionality. A silicon-based implementation may preclude the necessary scrutiny to which safety-critical systems should be subjected.

- There are no techniques and methods of design, documentation, testing, and verification identified or recognized by the FAA for today's complex hardware designs. Existing guidance does not address current practice or technology.

- There are failure modes associated with ASICs that are not readily identifiable. They can be the result of design errors or subtle phenomena that are not flagged by the tool suite, or discovered during device testing.

- ASICs incorporate design characteristics and techniques from both hardware and software disciplines. Issues relating to the hardware/software co-design process will need to be addressed in future avionics systems.

This report points out that in order to generate a large ASIC, tens of thousands of lines of code can be required. Large ASIC development has been described as a "software project being performed by hardware engineers" (Corcoran 1995). Not only are there the normal hardware integrity issues for safety-critical systems, but now, all the issues of software correctness apply also. The suite of computer-based tools and hardware descriptions which are generated with a software-based hardware description language are not currently regulated by any guidelines while tools used in software development are. Many of the issues that plague complex software systems are also emerging in complex integrated circuit (IC) designs.

1. INTRODUCTION.

While the term *complex* used in the report title is a subjective assessment, the authors take the position that an integrated circuit (IC) design is complex if it is impractical to manage without the assistance of design automation tools. In general, the industry consensus is that an IC design requires the use of design automation tools when it exceeds the ten thousand gate level. At the time of this report, technology has evolved to the point where user-programmable IC designs of a million or more gates are possible.

The topic of *Complex Integrated Circuits* was identified by a number of government and industry aerospace experts as one of many areas where certification specialists require more information and training. A familiarity with a technology is essential so that valid safety assessments of systems presented for certification can be made (Janowitz 1993). Based on the rapid growth of digital technology, a number of changes have taken place in the field of digital flight control and avionic systems that call for a close examination of the risks involved with the use of application specific integrated circuit (ASIC) technology.

In the past, the design of ICs was performed by IC manufacturers. ICs were produced that were generic enough to find wide application among digital designers. In the 1980s, new technologies emerged that allowed the designers to produce small custom ICs using programmable and reprogrammable parts that were readily becoming available. Levels of on-chip integration were also increasing, allowing manufacturers to offer devices with increasingly greater amounts of functionality. Using these parts meant that IC counts could be reduced, power reduced, inventories reduced, designs changed more easily, and products produced more quickly.

A significant step in the advancement of digital technology came with the introduction and rapid proliferation of ASIC technology. When originally introduced, ASIC development was expensive and ASICs were not widely used. However, rapid advances in ASIC technology coupled with major improvements in the design tool suite has caused an industry-wide shift which focuses on ASIC technology.

The continuous growth in the level of integration and in ASIC technology created a void for tools required to handle the tasks of ASIC design, test, and manufacturing. Traditional methods for digital design gradually disappeared while new and more efficient methods were developed. Digital hardware designers now face many complex issues: i.e., selection of a hardware programming language, tool suite selection, tool database translation, device testability, and submicron-related design considerations.

1.1 NEW THREATS FROM AN OLD TECHNOLOGY.

The use of digital technology in aircraft is nothing new. Implementations of early digital logic ICs were relatively easy to analyze and their failure modes were well understood. While the use of ASICs in aircraft is seen as a benefit for avionics manufacturers and airframers, their implementation raises concerns about the safety of systems in which they are used. Part of the problem is due to the sheer complexity of current ASIC devices. Failure analysis guidelines that were developed for digital systems, such as SAE's ARP1834, cannot be applied in a meaningful way to the complex ICs that are being designed today. Additionally, failure modes that were nonexistent with older digital technology are now prevalent and can compromise the safety of systems using these complex devices.

Commercial fly-by-wire aircraft are now being produced that have differing design philosophies from earlier aircraft. While digital avionics and flight controls have existed for a number of years, there were always backup systems that relied on different technologies in case of a failure of the digital system. These were hybrid aircraft, using a combination of control hydraulics with interfaces to digital systems. Today's fly-by-wire aircraft, as typified by the Boeing 777, use data buses to send actuation commands, based on messages generated from the avionic systems, to the various control surfaces. On these aircraft, the hydraulic link no longer exists, even for backup purposes. It is expected that if there is a failure on a primary system, another system with duplicate capabilities and connectivity will be available to take control. Back-up systems are simply duplicates of the primary systems. Redundancy may sometimes be implemented using dissimilar hardware and software and integrity enhanced by voting systems and other techniques, but the technology remains the same.

Due to increased IC densities, ASICs can now be programmed to take on tasks that were formerly performed in software. For instance, communication protocols are implemented in a chip. High-Level Data Link Control (HDLC), and ARINC 629 are complex transactions that are described in a written specification, and implemented in silicon. Issues that should be examined or verified include:

- The protocol is completely and correctly specified in written form.
- The implementer correctly understands the protocol.
- The protocol was implemented completely and correctly in silicon.
- The silicon is not defective.

These are nontrivial issues. At current ASIC complexity levels, it is dangerous to assume that upset avionics are due solely to software bugs.

A new urgency, therefore, exists to ensure that these digital avionic systems are designed correctly, implemented correctly, tested fully, and are reliable in every other respect. This is no trivial matter when one considers the current complexity level of ICs. Error-free parts can no more be guaranteed than one can promise error-free software. In fact, complex ASIC design is described by Corcoran (1995) as a "software project being performed by hardware engineers," since most ASIC parts are now designed using high-level languages that can describe digital logic behavior.

Complete testing of complex ASICs is not done since it is impractical. New testing techniques continue to be researched in order to discover new ways of enhancing an ASIC's testability. Even ICs, whose failure can cause substantial financial penalties to the manufacturer, such as the Pentium™, are not immune from process errors that can cause defective hardware.

There are a number of other areas of concern relating to the use of ASICs in flight-critical systems. They include:

- ASICs have been used to avoid the rigors of the software approval process (Shaw, Herzog, and Okubo 1986). Safety-critical functions have been implemented completely in hardware. Fundamental verification issues can be bypassed with a silicon-based implementation.

- Due to the current ability to create ASICs and Field Programmable Gate Arrays (FPGAs) using software (i.e., at a much higher level of design abstraction than in the past), systems engineers and even programmers are designing ASICs. Meaningful verification may not be possible since knowledge of all development steps and their products is necessary.

- ASICs can contain embedded microprocessor cores with user-supplied software. Complex ASICs can replace complete systems containing Programmable Read-Only Memory (for software memory), Central Processing Units, Digital Signal Processors, Random Access Memory, and other random logic elements.

- Due to their level of complexity, ASIC functionality cannot be completely simulated and the silicon completely tested.

- As avionic ASIC densities increase, it may become difficult for manufacturers to demonstrate that sufficient levels of testing have been performed to ensure device reliability.

- ASIC designs are implemented using a full suite of computer-based tools that are not regulated by any guidelines while tools used in software development are. (RTCA-DO-178B requires tool qualification for software tools in safety-critical applications.)

- Tool-induced design errors occur and can be difficult to detect. The fidelity and completeness of simulation tools is essential to design integrity. Designers are sometimes left without adequate tool support in some areas of design and test.

- It can be difficult to detect faulty operation of an ASIC. Complex ASICs require that test circuitry be designed into the ASIC. Designs for fly-by-wire aircraft require fault tolerant architectures, not simply redundancy. There are no guidelines that would suggest to airframers how this is done or what to require of their avionics suppliers.

- Due to extremely small ASIC geometries, certain analog and transmission line phenomena occur internal to the ASIC, generating failures that are data-sensitive. Often, designers and tools do not account for these effects and they can easily escape notice during test.

- There are failure modes associated with ASICs that are not readily identifiable. They may ultimately be the result of design error.

- Digital logic circuits are increasingly susceptible to upset by cosmic radiation as transistor densities increase (Keller 1993). As submicron ASICs are incorporated into avionic designs, methods of decreasing this vulnerability may be required.

- ASIC designers cannot guarantee error-free designs, and ASIC manufacturers cannot guarantee error-free silicon. As ASIC complexity increases, the likelihood of a latent fault also increases.

- There are no techniques and methods of design, documentation, testing, and verification identified or recognized by the Federal Aviation Administration (FAA) for today's complex hardware designs. A formal process for developers and manufacturers of avionics needs to be defined, because existing guidance does not address current practice or technology.

ASICs typically are programmed by the end user or avionics supplier. Complex ASIC designs require teams of 50 to 100 engineers. ASICs are a technology essentially unregulated by the FAA and not understood by certification engineers (CEs). The level of on-chip circuit elements that can be squeezed into an ASIC is so high that more of the software portions of avionic system designs are being placed into ASICs. ASICs are used extensively on the Boeing 777 in flight-critical systems and, along with other user-programmable logic and special function ICs, will be used almost exclusively in future fly-by-wire aircraft.

1.2 REPORT OVERVIEW.

In light of the many concerns relating to the use of ASICs in airborne avionics, this report will address current user-programmable IC technology from several aspects. Included in this report will be:

- Tutorial information which will provide a foundation for understanding the problems that are unique to the certification of systems employing complex ICs including design tools, design environments, languages, logic synthesis technology, simulation tools, and testing methods for complex ICs.

- Identification and examination of current standards and guidelines used in the development cycle of complex ICs.

- Examination of safety issues and identification of certification risks relating to the use of ASIC technology.

While the report can be read by all, it will be most meaningful to those with some understanding of digital logic. Numerous references are included in the Bibliography for further information on this and related topics.

2. USER-PROGRAMMABLE INTEGRATED CIRCUITS.

Initially, systems were designed using discrete logic ICs. These ICs were of varying complexities and included small-scale integration (SSI), medium-scale integration (MSI), and large-scale integration (LSI). Designers had no avenue for customizing circuits. If a particular function was unavailable on an IC, then it was implemented using as many discrete ICs as were necessary to provide the functionality.

Discrete ICs are typically easy to use. The ICs are fully debugged by the manufacturer. When considering only device cost, they are perceived as being quite inexpensive. There are, however, a number of drawbacks with the use of discrete ICs. Some of these are

- many ICs can be required, even for implementing small designs. FPGA technology may be a better choice for lower production quantities.
- design can be tedious at the level of detail required for discrete design.
- changes are difficult to make since a number of different parts may be affected when even small changes are made.
- designs can be difficult to debug.
- maintenance is more difficult.
- documentation costs are higher.
- since changes are more involved, design cycles are longer. Changes affect the schematic, printed circuit board layout, testing, and documentation.

While discrete ICs are still available from the manufacturers, they are rarely used in new designs. They are still commonly used for bus drivers and line drivers but are contained in much smaller packages than the Dual In-line Packages (DIPs) in which they were originally available. Due to the reasons mentioned above and constant product improvement initiatives that make systems operate faster, use less power, and take up less space, use of SSI, MSI, and LSI devices has fallen off dramatically. Additionally, design cycles are constantly being shortened, calling for a design environment and tool set that is completely different from what has existed in the past.

2.1 PROGRAMMABLE LOGIC DEVICES.

In general, a programmable logic device (PLD) is an IC that is configured by the user to perform a particular logic function, or combination of functions. Each type of PLD has a unique set of features that have advantages and disadvantages for each type of design. Cost-effective designs that can meet task requirements can be realized only if the designers account for the unique advantages and disadvantages of the various PLDs and their characteristics.

Early ICs, consisting of 14- and 16-pin SSI logic, did not allow designers any flexibility to configure their own logic. Programmable Array Logic (PAL), introduced by Monolithic Memories in 1978, is regarded as the first PLD. It used programmable read-only memory (PROM) technology to allow users to write patterns into the PAL, which configured the internal logic according to the predefined device definition provided by the manufacturer.

PLDs benefit applications where the design is expected to undergo numerous changes. Typical gate counts for these ICs run from the 100s to around 20,000. Designers make choices between PLDs based on reprogramming requirements and cost factors.

Two basic types of PLDs are in common use. One contains on-chip memory storage of interconnect patterns that are both erasable and reprogrammable. The other is based on a link or fuse technology and can be programmed only one time. Reprogrammable devices allow designers to use the same IC type for several portions of the design as well as allowing product upgrades without having to change the printed circuit board design.

The tool sets required to develop PLDs are typically inexpensive to purchase, easy to understand and use, and run on low-cost personal computer (PC) platforms. Tools for PLDs can handle input from schematics or hardware description languages (HDLs). It is expected that continued advances will be made in PLD technology which will allow PLDs to overtake more of the low-end market share of more complex devices such as gate arrays.

A PLD's interconnect method reflects upon its performance capability. Interconnects occur in two ways: the crosspoint switch configuration and multiplexer-based approach. In the crosspoint switch configuration the interconnects are fully routable. It is possible to route any combination of internal signals to or from any logic block. A drawback of this method is that a large amount of silicon is consumed and signal speed is also reduced for the interconnects.

The multiplexer-based approach distributes all signals through multiplexers. This method reduces the die area required for interconnections and also enhances device signal speed. There are a number of ways used to implement this technique, each technique having different characteristics for performance and routability.

Routability can often be a limiting factor in PLDs. If the maximum number of signals that can be routed to a logic block is exceeded for a particular logic function, then the design will need to be partitioned among other free logic blocks. It can also be difficult to find an interconnect that routes all the required signals into the appropriate logic blocks. As the number of signals that are routed into a logic block approach a maximum, it becomes much more difficult to route the remaining required signals.

An additional problem occurs when there are design changes after a device has gone through a successful layout and pinout configuration. A design may no longer fit with the same pin configuration and the result may be considerable rework for the designer. Hence, routability is an important consideration in the choice of a suitable device.

Characteristics of the individual logic blocks are an important consideration in the use of PLDs. The amount of logic contained in a logic block and the flexibility of these blocks to accommodate different configurations have given designers more design possibilities. One popular logic device which was designed years ago is the 22V10. In it, the number of product terms available in a logic block is not variable. If there are terms that are not used, they end up wasting space since they cannot be steered to another logic block for combining with the terms of that block. Also, the logic blocks of this device do not share product terms. Therefore each logic block that requires a particular term must generate it completely. Efficient utilization of logic block resources and performance enhancement can be achieved through sharing and steering.

As improvements in the design of logic blocks progressed, vendors began to offer more options. These included steering and sharing of product terms. Product term steering allows designers to move a portion of the logic block from one cell to another. While this helps a cell that requires more functionality, the cells from which the functionality was removed are now reduced in usefulness. This lack of functionality may then result in wasted resources.

Product term sharing allows logic to be implemented once and used in multiple logic cells. This allows the generation of more complicated functions since the terms are, in essence, cascaded. One factor to keep in mind is that product term sharing, due to the cascaded connections, can introduce unacceptable delays into the design. Through internal architecture changes, some of the more advanced PLD designs allow steering and product term sharing without the delay problem or wasted resource condition (Kapusta 1995).

Many other variations of PLDs have been developed and are now available to designers. Some of the numerous changes that have taken place in PLD technology include

- diverse architectures, allowing developers a variety of functional design implementations;
- reprogrammability based on static random access memory (SRAM) provides for rapid prototyping and testing of new designs as well as for rapid logic reconfiguration at the device SRAM speed;
- changes in the implementation technology, reducing power consumption and operating supply voltage and increasing clock rates;
- proliferation of tools which enhance the designer's productivity; and
- major increases in the level of on-chip logic, package size, and the number of device input/output (I/O) connections.

The following sections contain information intended to familiarize the reader with user-programmable ICs, various architectures, and possible implementations.

2.1.1 Programmable Array Logic

Boolean expressions are commonly used to represent and program logic contained in PALs and other programmable logic devices. Basic combinatorial logic elements are shown in figure 1.

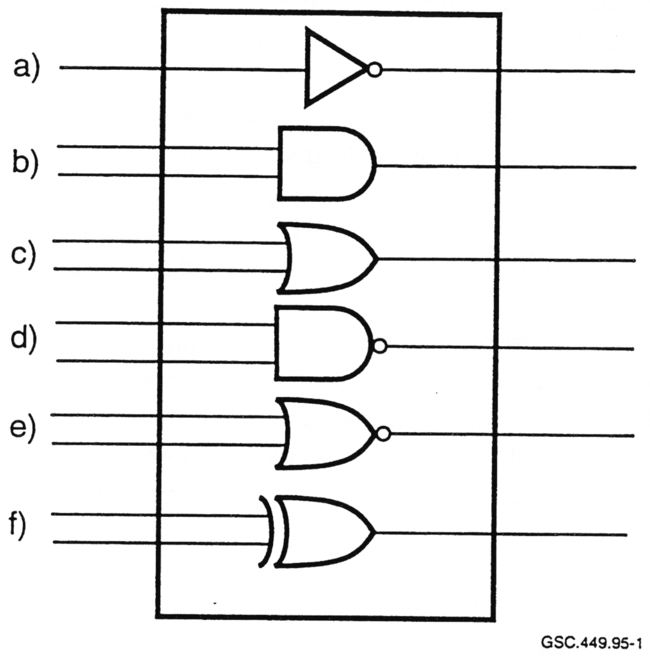


FIGURE 1. BASIC COMBINATORIAL LOGIC ELEMENTS
(PAL Device Data Book and Design Guide 1995)

Figure 1 (a) is a logic inverter. Its logic function is represented by

B = /A

where A is the output, B is the input, "/" represents the compliment of the signal, and "=" is the assign character. This equation means that B is the compliment of A. A listing of combinatorial functions shown in figure 1 are found in table 1.

The earliest PAL devices consisted of an array of AND-OR logic similar to that shown in figure 2. Input signals are along the left side and can be programmed in the complement or true (not complement) state. Each of the these input signals is then connected to one of the vertical "input lines." These input lines are then selectively connected to the horizontal lines to form product terms. The product term is the AND of all input lines that are connected to a particular horizontal line. Since it would be quite cumbersome to represent all the AND input terms as a separate input to the AND gate, this shorter form of representation is used.

TABLE 1. COMBINATORIAL LOGIC FUNCTIONS

Figure Reference	Equation	Gate Name
1 (a)	B = /A	NOT Gate
1 (b)	A = B * C	AND Gate
1 (c)	A = B + C	OR Gate
1 (d)	A = /(B * C)	NOT AND (NAND) Gate
1 (e)	A = /(B + C)	NOT OR (NOR) Gate
1 (f)	A = B * /C + /B * C or A= B :+: C	Exclusive OR (XOR) Gate

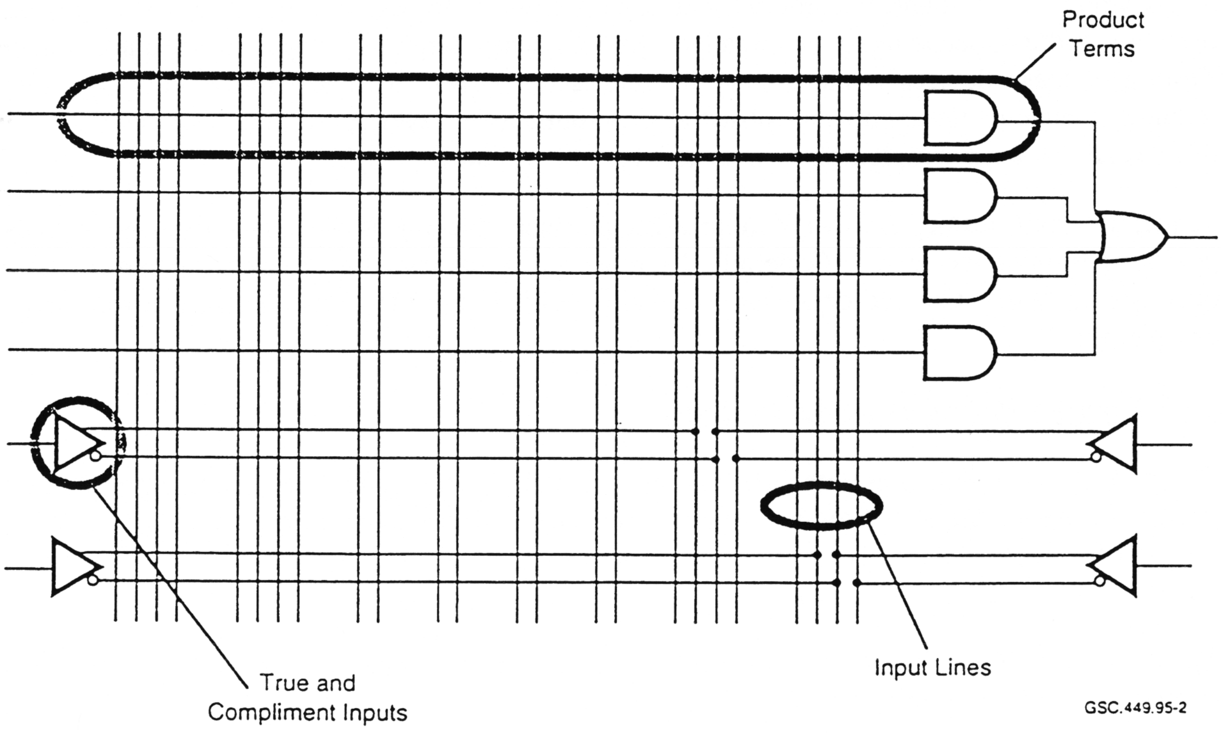


FIGURE 2. SECTION OF PROGRAMMABLE ARRAY LOGIC
(PAL Device Data Book and Design Guide 1995)

The designer selects which of the input lines to include in each AND term. This is typically done using a software-based tool. The tool translates simple logic equations expressed in an AND-OR format that are associated with each group of product terms the designer desires to implement.

In the figure, the OR inputs are fixed by the connections that are made to the four AND terms. This type of array logic is configured for decoding combinations of binary patterns, such as addresses. Output pins for the AND-OR terms are generally assigned by the designer.

For instance, if the designer wanted to decode the hexadecimal inputs 01H, 02H, and 04H, these terms would be expressed as follows:

!A!B!CD + !A!BC!D + !AB!C!D

where "A" represents the most significant bit and "D" the least significant bit, "!" represents the NOT function, and "+" represents the OR function.

When the equations are compiled by the tool, the tool produces a file designed for a specific target IC. This file is in one of several standard formats that allow downloading to a device programmer. PALASM™,CUPL™, and ABEL™ are some of the common software packages that support many of the common programmable ICs available.

PALs reduce the amount of combinatorial logic necessary for decode functions. They are often used for address decoding or as small controllers when sequential logic is included on-chip. Testing of these devices is elementary if there is no sequential logic. In-circuit testing simply requires control of the inputs and observable outputs, along with appropriate test vectors. These devices are the least complex of the available user-programmable logic.

Sequential logic testing is always more difficult since clocking is necessary to cause signals to propagate through the device. Generally, the more sequential logic on a device, the more difficult it becomes to test. Correct operation is easily verified on PALs due to their lower level of complexity.

2.1.2 Complex Programmable Logic Device.

Complex PLDs (CPLDs) are essentially a number of PLDs integrated into a single programmable device. Some of these devices are being manufactured using submicron technology. Similar to other PLDs, these devices contain blocks of logic which are interconnected by a programmable matrix.

The number of logic blocks can vary, allowing designs to range from simple to complex. A number of other parameters can vary, including:

- I/O pins, up to 200 or more,
- logic macrocells, up to 256,
- flip-flops, up to 300, and
- clock speed, up to 150 Mhz.

These figures will keep increasing as the technology continues to develop.

The programmable interconnect matrix serves as a global signal router for signals from the I/O pins and the logic block feedback signals. It is possible to route any signal from any pin to any or all logic blocks. A block diagram for the CPLD is shown in figure 3.

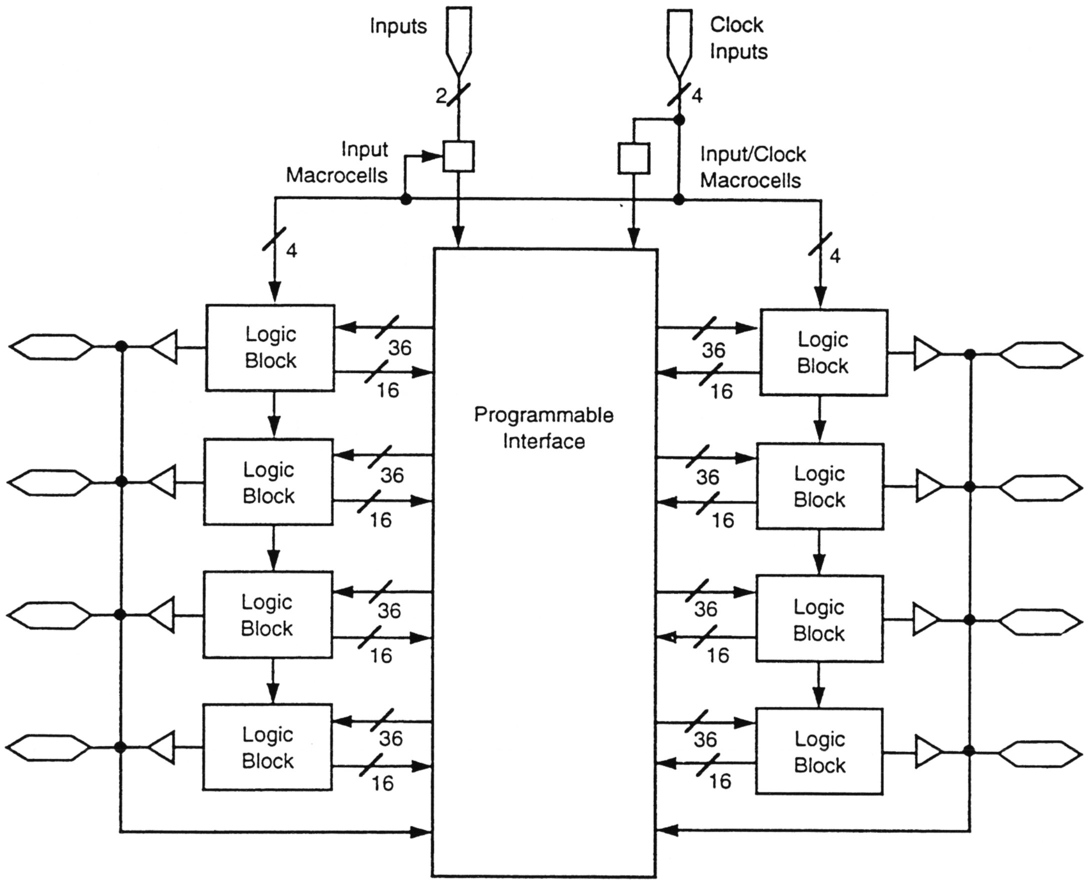


FIGURE 3. COMPLEX PROGRAMMABLE LOGIC DEVICE BLOCK DIAGRAM

Propagation delays through the interconnect matrix are already accounted for in the timing specification for each device. All inputs travel through this matrix. Also, there are no route dependent timing parameters on these devices. All routing for the interconnect matrix is handled by the development software. Manual routing is not done and the software handles all routing in a matter of minutes.

The logic block is the basic building block of the CPLD. The logic block contains a product term array, a product term allocator, a variable number of I/O cells, and 16 macrocells. Figure 4 shows the basic elements of the logic block and their interconnections.

The logic block can be chosen as register-intensive or I/O-intensive. For the I/O-intensive architecture, there exists an I/O cell for each macrocell. In the register-intensive architecture, there are eight macrocells that are connected to the I/O cells. There are also eight internal registers that are fed back into the interconnect matrix. This allows more design flexibility, especially for register-intensive applications.

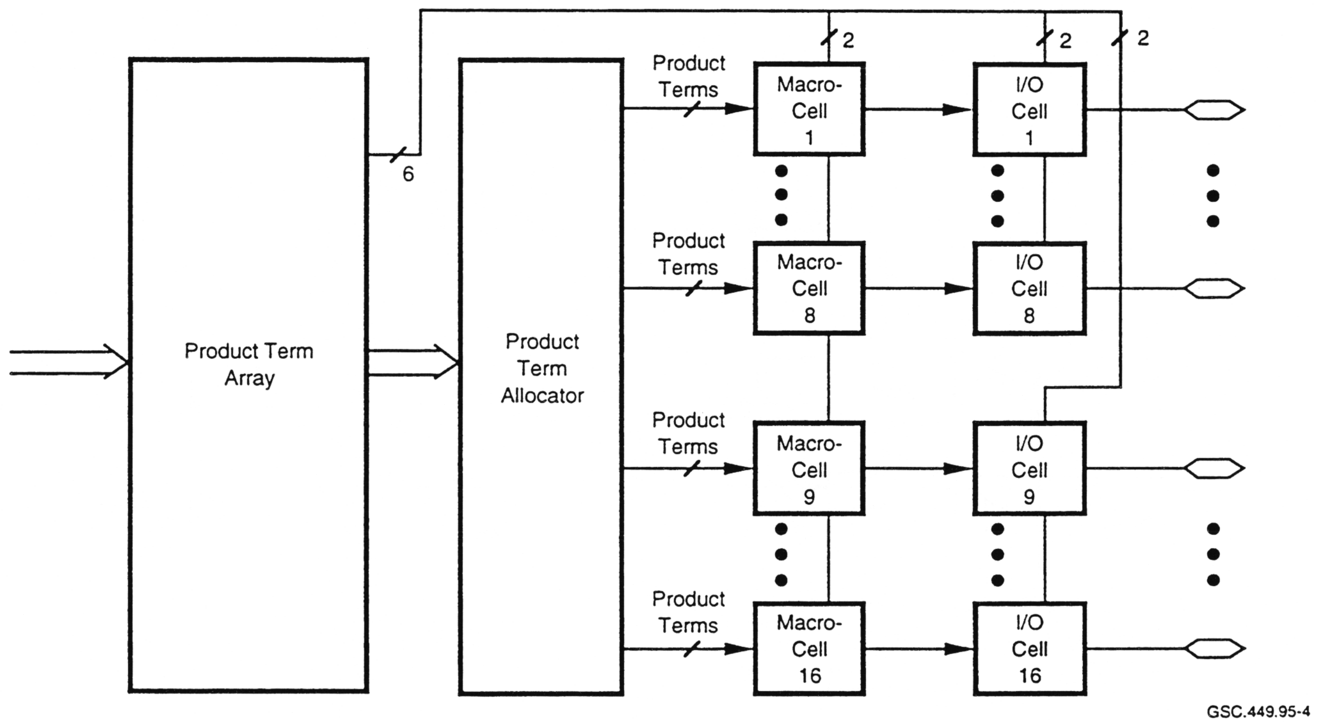


FIGURE 4. BASIC LOGIC BLOCK ELEMENTS

Apart from architectural considerations, one of the most important features is knowing the device timing. Worst-case timing delays need to be considered. Additionally, designers should note whether or not timing is influenced by how the circuit is implemented. Also, if design performance is influenced by the device programming, designers need to know how. Among the factors that can influence device timing are:

- routing delays,
- product term sharing delays,
- fan-out delays,
- interconnect matrix delays, and
- other architecturally unique device delays.

CPLDs can serve to reduce the amount of random logic which is necessary on many designs. They can replace up to 200 or 300 SSI devices. They do not contain any built-in logic for testing. Once a device is programmed and in the target environment, there is nothing that can be done for device testing in-circuit unless it has already been designed into the device.

CPLDs can be used to implement complex state machines. They may easily execute control algorithms containing more than 200 states. Most of these devices are also capable of very high-speed operation. For this reason, they often are used for applications requiring high-speed controllers, such as bus interfaces.

2.1.3 Erasable Programmable Logic Device.

Recent advances in erasable PLDs (EPLDs) have yielded devices with greater performance and flexibility. They are programmed and reprogrammed using erasable PROM (EPROM) technology. A Xilinx Incorporated EPLD that became available in 1995 boasts the following features:

- 7.5 ns propagation delay,
- up to 3500 usable gates,
- 24 mA output drive capability,
- I/O interface accommodates 3.3 V or 5.0 V systems, and
- 160 or 225 pin versions.

EPLDs allow for LSI levels of integration. They contain multiple programmable logic structures (function blocks) that are interconnected in a large matrix. Each of the function blocks contains a number of macrocells which have AND/OR arrays for input. Figure 5 shows a block diagram of how the EPLD is connected.

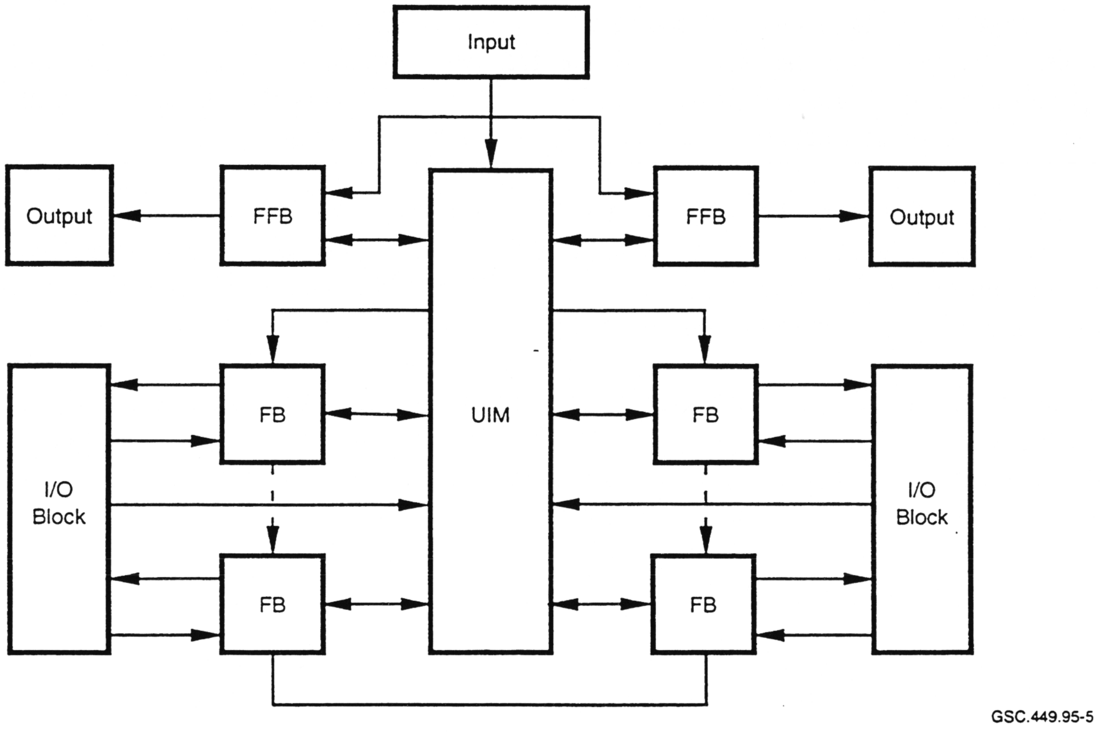


FIGURE 5. EPLD BLOCK DIAGRAM
(Programmable Logic Data Book 1994)

This particular EPLD has an architecture that allows not only high-speed operation using Fast Function Blocks (FFBs) but also high-density capability with high-density function blocks. FFBs are used where pin-to-pin delays need to be minimized. This may be required for high-speed decoding or very fast-state machine designs. More complex functions can be implemented in the high-density function blocks.

An interconnect matrix is located in the center. Complimentary metal oxide semiconductor (CMOS) EPROM technology implements the interconnect matrix. This matrix ensures the full interconnection of all design functions. Ideally, the matrix should provide a minimal, non-varying delay between points so that device timing is predictable. The interconnect matrix receives inputs from each macrocell output, dedicated input pin, and I/O pin. It generates 21 output signals to each high-density function block and 24 output signals to each FFB. Each input can be programmed to any output.

Inside each function block there are nine macrocells. The construction of a macrocell is shown in figure 6. The flow of data within the macrocell is from the left to the right. While many of the input signals may not be used in a typical design, the fact that they exist gives the designer great flexibility in the types of circuits that can be implemented. The various inputs are directed into the programmable AND array with the exception of two of the inputs. These control the output enable function. The particular family of devices based upon this macrocell uses a universal interconnect matrix (UIM) which enables global signal routing between macrocells.

The core of the logic consists of a section that is configurable as a D-type flip-flop, toggle flip-flop, or transparent latch. Inputs to this section are from the AND array along with other user-selectable controls. Outputs from the AND array are fed into a multiple input OR gate, allowing for computation of a sum-of-products term for this macrocell. There is also an input from the previous macrocell's sum-of-products term, and a similar output to the next macrocell.

The high density function blocks are slightly different from the FFBs. An arithmetic and logic unit (ALU) is included as the logic core. The ALU can be programmed to make either arithmetic or logic calculations. Included in this portion of the function block are carry lookahead capability for the ALU and a flip-flop on the output of the ALU.

Since the EPLD is EPROM-based, it lends itself to use in a development environment where physical design iterations are required. The EPROM array which configures the device is erased by exposure to ultraviolet light of the correct intensity, wavelength, and time duration. Opaque covers are placed over the EPROM windows to prevent erasure.

2.1.4 High Capacity Programmable Logic Device.

The High Capacity Programmable Logic Device (HCPLD) is yet another variation in the PLD market. The PLD market began in 1984 with a 1,000 gate device. Decreased cost, increased speed, and current devices with up to 50,000 gates have made the HCPLD attractive to circuit designers. Used originally to replace "glue logic," they are now being used to prototype or replace gate arrays (Beechler and Hanz 1994).

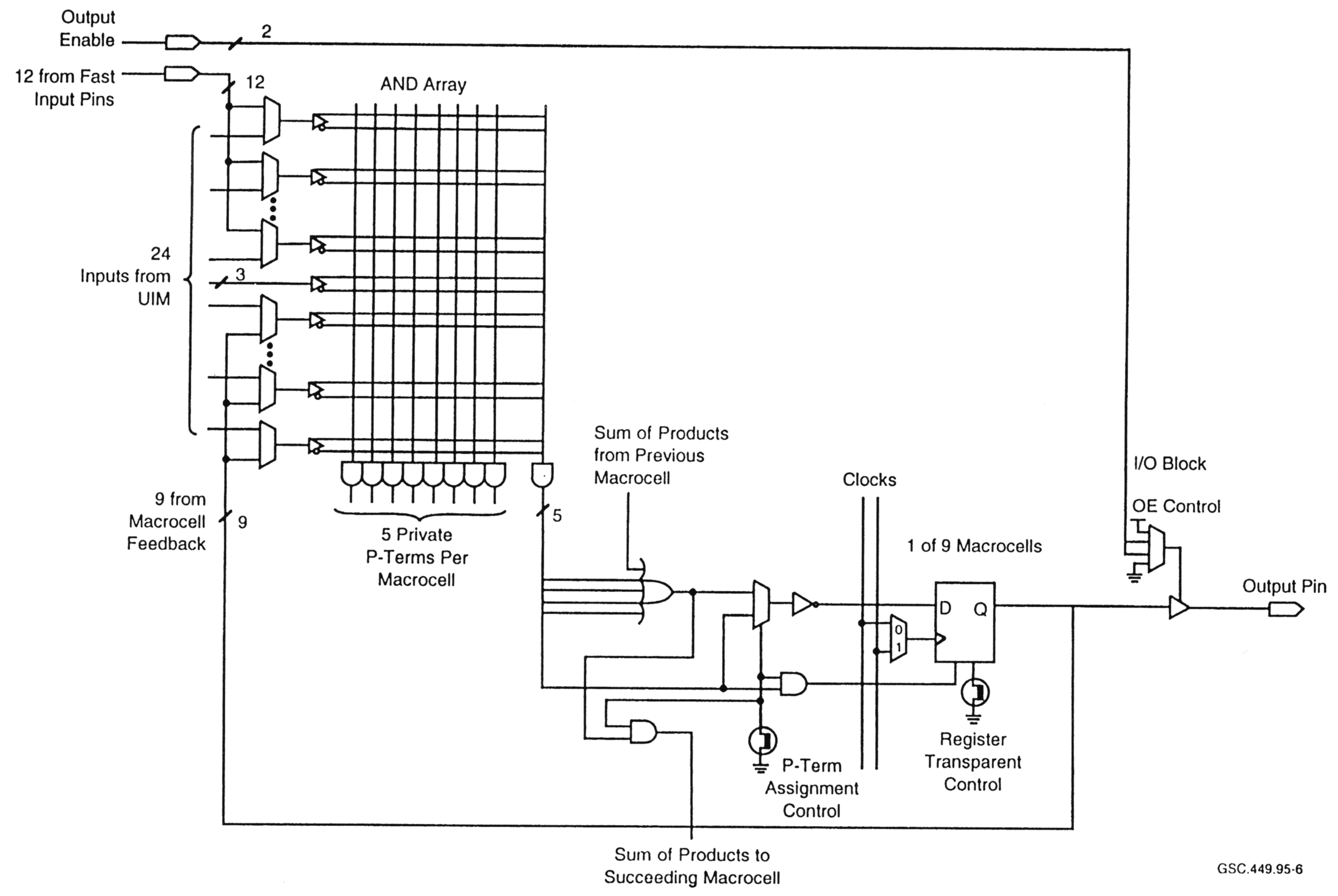

Output Enable
2
12 from Fast Input Pins
12
AND Array
24 Inputs from UIM
3
9 from Macrocell Feedback
9
5 Private P-Terms Per Macrocell
5
Sum of Products from Previous Macrocell
Clocks
I/O Block
OE Control
1 of 9 Macrocells
D
Q
0
1
Output Pin
P-Term Assignment Control
Register Transparent Control
Sum of Products to Succeeding Macrocell
GSC.449.95-6

FIGURE 6. BASIC MACROCELL CONSTRUCTION
(Programmable Logic Data Book 1994)

2.1.5 State Machine Integrated Circuit.

The state machine is a design technique used for modeling software as well as hardware execution flows. For hardware, it is commonly used for modeling protocols and bus control interaction. The state machine is also used to implement complex logic control. Synthesis tools decompose HDL code into smaller modules that are implemented as state machines.

In the past, designers have been creative in their construction of hardware state machine designs, lacking off-the-shelf support components. Now, EPROM ICs specifically designed for use as state machines are available. One device, in particular, can be programmed with over 16,000 discrete states. These devices contain other logic in addition to the contents of a typical EPROM. The additional logic includes input and output latches, input multiplexers, and a feedback path from the device output signals to the input of the multiplexer.

High-level state machine design specifications and microprocessor software can be expressed in similar terms by using flow-charting. The difference between the two technologies comes in the added flexibility and programmability of central processing unit (CPU) software. A CPU is, in essence, a special purpose state machine with added flexibility.

2.2 APPLICATION SPECIFIC INTEGRATED CIRCUIT.

2.2.1 Overview.

ASICs are divided into two categories: the gate array and the standard cell. These names refer to the circuit configurations that exist within the ICs. Gate arrays can be one of two types: field programmable or masked. The field programmable version is the most complex of all PLDs. Figure 7 gives a simple block diagram of the gate array configuration.

The internal configuration for the gate array consists of a large array of identical logic blocks. The basic circuitry is prefabricated and requires programming the interconnects based on the design requirements. The gate array lacks the features necessary for customizing circuits but excels in that the logic that exists can be programmed rapidly. For the gate array, the basic die is prefabricated. Only the interconnects between and in the logic cells and the connections to external pins need to be made.

The standard cell configuration, as shown in figure 8, also is comprised of blocks of logic. However, the blocks are not all the same and they are not laid out in a uniform and repetitive pattern as are gate arrays. The basic die has not been prefabricated and layout of a complex ASIC can be difficult. Minimization of the number and length of on-chip interconnects is a major goal in IC design and layout in order to reduce the likelihood of layout-induced problems.

A comparison of gate array and standard cell designs is given in table 2. As indicated in table 2, both the gate array and the standard cell devices have significant advantages depending on the overall application. In the gate array, where the logic blocks already exist on-chip, the nonrecurring engineering costs are lower; hence the turnaround time is faster. Also, the off-the-shelf cost of a gate array is higher than for a standard cell part. The standard cell should be used where higher device quantities are required. They have the advantage of being highly customizable since the manufacturer provides predefined libraries of common functions for the designer. Utilization of the silicon area is high given the freedom to implement only those circuit portions that are required for the design.

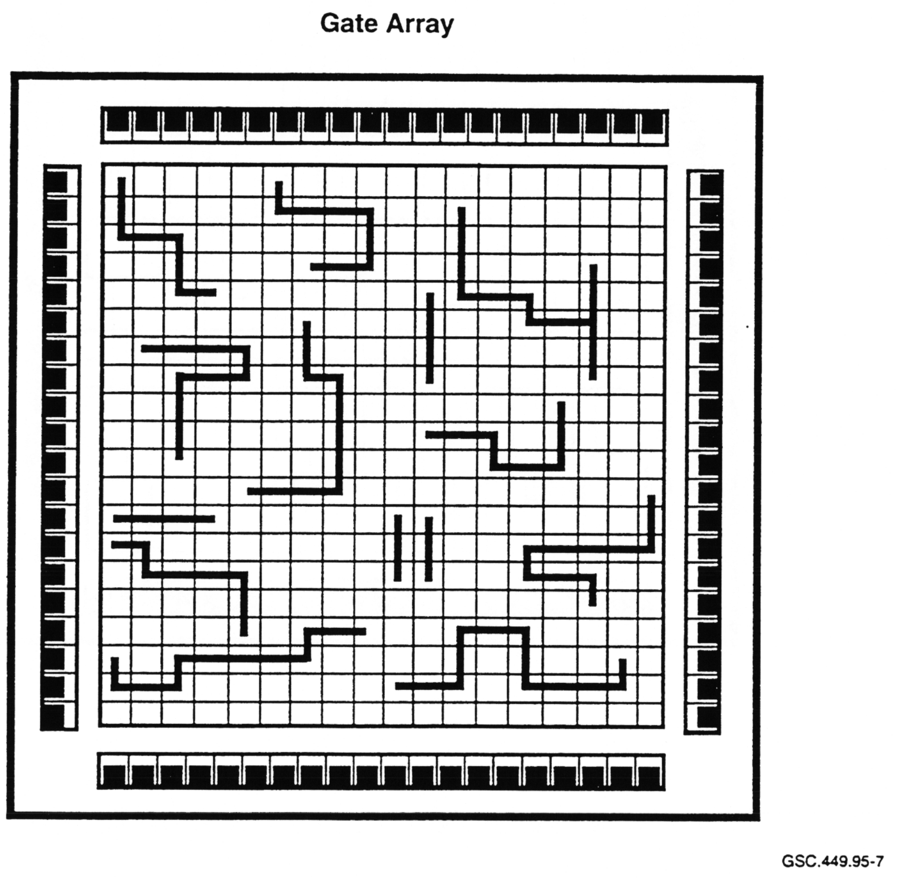


FIGURE 7. GATE ARRAY BLOCK DIAGRAM

TABLE 2. GATE ARRAY AND STANDARD CELL COMPARISON

Characteristic	Gate Array	Standard Cell
Nonrecurring Engineering Cost	Low	High
Per Piece Cost	High	Low
Utilization	Low	High
Turnaround Time	Fast	Slow
Customizability	Low	High

Standard Cell

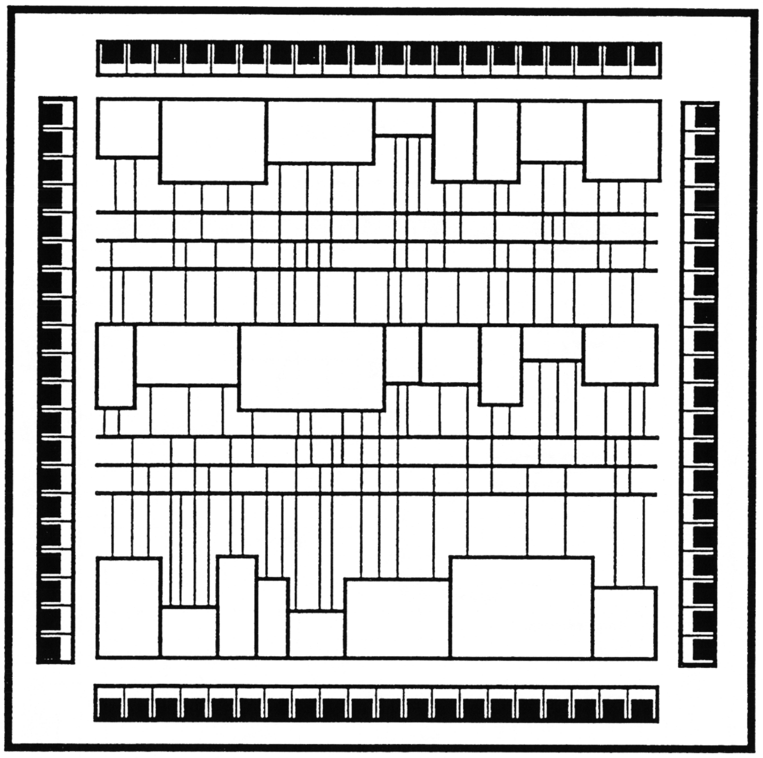

GSC.449.95-8

FIGURE 8. STANDARD CELL BLOCK DIAGRAM

2.2.2 Field Programmable Gate Array.

Compared to other types of programmable logic, the gate array offers the designer more flexibility and options. Each IC is customized for a particular application by programming the internal interconnections. These interconnections consist of vertical and horizontal routing conductors that are used to connect the various internal gate array function blocks and I/O circuitry. An FPGA logic cell is typically smaller and less complex than the cell of a CPLD. A cell normally consists of a register, associated logic, multiple inputs, and multiple outputs.

When designs are required that are too complex for other types of PLDs, gate arrays often provide designers with the necessary solution. Also, if large volumes of PLDs are required, migrating the design to a gate array can cut the production costs. Currently, usable gate counts for gate arrays can be in excess of 200,000.

It can be difficult and challenging to utilize large portions of the available silicon in a gate array design since there are portions of logic blocks that are not needed by the designer. There is no way to eliminate these and reclaim the space. If a more complex design is necessary than will fit into a gate array, the designer can use a more complex gate array or migrate the design to a standard cell implementation.

Increases in gate count and ease of programming these devices has led to increases in the use of FPGAs. Ability to design and program parts on the desktop personal computer (PC), design portability, and FPGA-specific tool support such as synthesis and simulation allow designers to increase productivity by using FPGAs as compared to other types of PLDs.

2.2.2.1 Reprogrammable Field Programmable Gate Arrays.

One of the more interesting developments in FPGA technology is SRAM-based reprogrammability. Typically, once the FPGA is programmed, any changes will require the use of a new and unprogrammed IC. However, the SRAM-based FPGA allows reuse of the same device when logic changes are necessary. Also, there is no limit on the number of times that an SRAM-based FPGA can be reprogrammed.

Essentially, SRAM-based reprogrammability allows users to develop soft hardware. With soft hardware, the user does not even need to remove the FPGA from the circuit in order to reprogram it. It can be reprogrammed externally while still operating in the target system. This reprogrammability allows the designer to modify a small portion of the FPGA's logic or the entire device.

The SRAM-based FPGAs are available from a number of manufacturers which all use the same basic concept of implementation. Internal static memory cells store the configuration program data that define the logic functions and interconnections. Rather than the traditional array structure, memory cells are distributed throughout the IC. In order to produce an efficient layout, each memory cell is located close to the logic which it controls.

Using a SRAM-based interconnect, the configuration must be loaded on each power-up. Additionally, if incorrect operation is suspected due to corrupted memory, the configuration must be reloaded. The configuration data can reside on an EPROM or disk. It is possible to design on-chip control logic to load the data directly from the external source without the use of a CPU. New memory patterns are typically loaded in a few milliseconds.

2.2.2.2 Field Programmable Gate Array Performance.

Signal delays in the FPGA can be caused by several factors, including:

- signal wire characteristics,
- programmable elements,
- amount of cascaded logic cells, and
- propagation delay of each logic cell.

The FPGA technology influences both the interconnect delays and routability. The two most common technologies used for interconnects are SRAM and antifuse. The antifuse interconnect is smaller than the SRAM and permits many more programmable elements to fit on a die. The SRAM-based FPGA is more restricted in routability. However, SRAM reprogrammability is a considerable benefit and a characteristic that some applications may require.

Interconnect net delays for FPGAs can be significantly longer than standard cell interconnect net delays. This is due to both the signal routing architecture and the crossing points, where the signal routes connect. These points can add a significant amount of resistance to the signal networks, which in turn will cause delays in the signal propagation.

Logic cell architectures also vary for SRAM- and antifuse-based FPGAs. Antifuse FPGAs generally use a fine-grained approach to cell structure which produces smaller and simpler logic cell structures. More logic cells are required to build a function with fine-grained cells than with coarse-grained cells. A fine-grained approach, therefore, relies more on the interconnect structure and cascaded cells than does the coarse-grained approach. Since this is the case, designs that use the fine-grained approach are more likely to exhibit slower performance. SRAM-based designs must minimize interconnects due to higher delay and a smaller fuse availability (Kapusta 1995).

During the early part of the design cycle, signal delays due to logic implementation and signal routing are largely unpredictable. Synthesis tools can make estimates based on a statistical wire delay model, but the values will invariably change in the actual IC. Where high speed timing is required, the accuracies of the statistical wire delay model may not suffice.

Limitations of the synthesis tool in accurately predicting the actual timing and the layout effects on timing may lead designers to be conservative in device timing estimates. Timing-related failures may appear later in the design cycle if tight margins are chosen initially. Without holding to tight margins, however, designs become less competitive.

2.2.2.3 Basic Cell Configuration.

The basic cell composition for the gate array formerly contained the transistors necessary for implementing two or more gates per cell. Now, the basic cell is optimized to build a flip-flop or random access memory (RAM) element. It is much easier for designers to build required gates from flip-flops than it is to build flip-flops from gates.

2.2.2.4 Logic Block Configuration.

Logic blocks can vary in their implementation. One possible configuration is shown in figure 9. Note the massive amount of interconnects that are possible. This large matrix gives the designer great flexibility in the type of circuit that can be implemented. Signal flows are from the left to the right. It can be seen that all output signals from each gate or register can be fed back into the interconnect matrix for implementing the next-state logic for the register or for use elsewhere within the device.

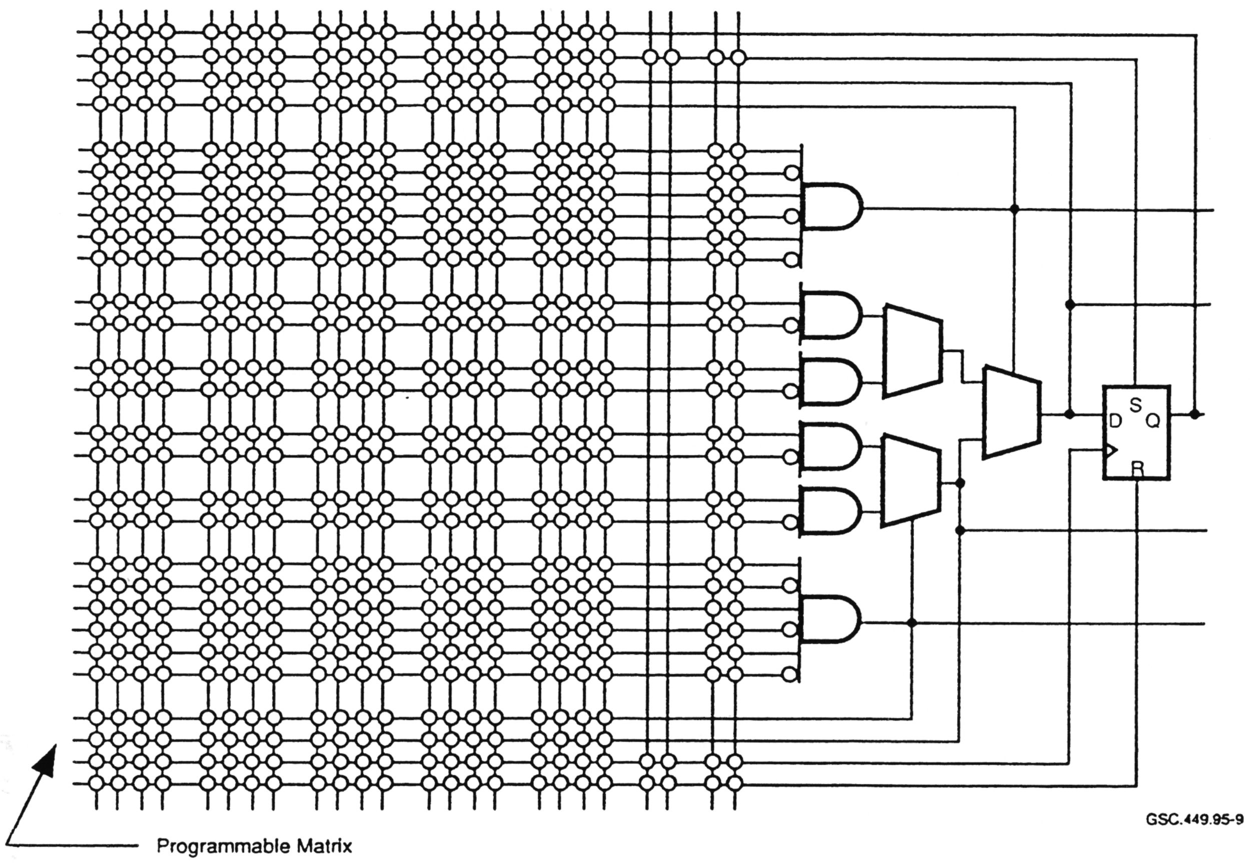


FIGURE 9. GATE ARRAY LOGIC BLOCK

2.2.2.5 Basic Design Cycle.

Fawcett[3] (1994) divides the design cycle for FPGAs into three steps: design entry, design validation, and design implementation. This concept is shown in figure 10.

Design entry involves specifying gates, registers, and interconnections. This is generally done by either creating a graphical representation on paper or computer or by using an HDL. Validation tests the final design to check if it performs its intended function. It may use timing analysis and system-level simulation among other methods. Design implementation involves mapping the design into the chosen FPGA, placing the logic resources, and picking an optimal interconnection scheme based on this placement.

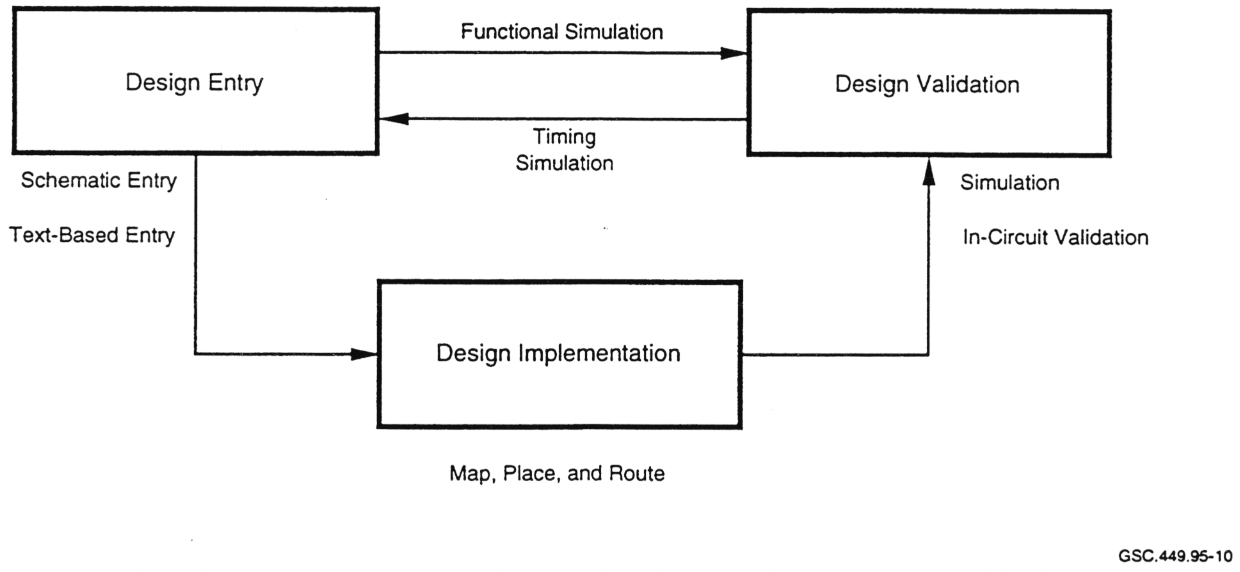


FIGURE 10. FIELD PROGRAMMABLE GATE ARRAY DESIGN CYCLE (Fawcett[3] 1994)

2.2.26 Advantages and Disadvantages of FPGAs.

Advantages of FPGAs include:

- FPGA use can provide greater device reliability over multiple IC implementations of the same design.
- Higher densities are possible for FPGAs than with other types of PLDs.
- The use of HDLs is supported in order to manage the design complexity.
- FPGAs allow greater architectural flexibility, such as allowing designers to implement SRAM cores.
- FPGAs are more cost-effective than standard cell ASICs for lower quantities.
- Scan logic can be implemented for easing testability.

However, there can be some drawbacks in the use of FPGAs. These drawbacks include:

- Development cost is higher, compared to other PLDs.
- FPGAs are not as easy to use as other PLDs due to their higher level of complexity.
- FPGA design may require the use of an HDL in order to manage complexity as opposed to the simpler logic representation expressions required for less complex PLDs.

- Since the unit cost is high, volume production is typically done with ASICs.

- Device timing is more difficult to handle for FPGAs than PLDs since, in general, fixed delays are not provided.

2.2.3 Standard Cell.

Designers now use ASICs routinely. Designers benefit when using ASICs for nonstandard logic implementations and high-volume production. Only in certain cases, such as highly cost-sensitive designs, are alternatives considered. When they are, a designer may turn to off-the-shelf or full custom components.

What distinguishes the standard cell from the gate array is that the complete mask is user-defined for the standard cell. This technology also uses full-design libraries of standard logic elements and memories. ASICs can contain combinatorial and sequential logic as well as analog circuitry. Logic forms that are implemented in the standard cell ASIC include:

- RAM,
- read-only memory (ROM),
- random logic,
- microsequencers and state machines,
- CPUs, or lesser portions thereof,
- fuzzy logic controllers,
- digital signal processors, and
- analog and mixed signal functions.

Standard cell library elements are optimized for either high speed or high density. Clock speeds for standard cells can run to several hundred megahertz and densities are in the hundreds of thousands of gates. The design cycle time for standard cells has traditionally been longer than that for gate arrays, but new tools are minimizing that difference.

Barriers for using ASICs have largely disappeared. Tools are available to run on all common engineering platforms.

Another consideration is the breadboard. With the complexity of ASICs, a breadboard development and test cycle would be prohibitive not simply due to the high pin counts, but because of the manually intensive testing that would be required. The proliferation of programmable devices, simulators, and other support tools such as in-circuit-emulators has, to a large degree, reduced the amount of breadboard activity.

3. DESIGN ISSUES FOR APPLICATION SPECIFIC INTEGRATED CIRCUITS.

3.1 DESIGN PHILOSOPHY.

Synthesis tools are designed to create efficient logic designs. They use a high-level description of the design and produce the low-level circuit design expressed as a netlist of gates and interconnects.

They run sophisticated algorithms and can tailor a design for speed or size. However, if the HDL code is cumbersome and not efficiently designed, then there is little that a synthesis tool can do to make it more efficient.

However, currently available tool sets are only beginning to address the system-level design. This aspect has become increasingly important to ASIC designers since submicron design technology has created the potential for complete systems on a single ASIC. Addressing system-level design involves techniques that apply not just to digital logic but to software as well.

3.1.1 Top-Down Design Methods.

ASIC designs need to be performed from the top to the bottom with knowledge of all issues from the bottom to the top. The requirements must be well-defined. Available tool sets and their capabilities must be known as well as available target devices and technologies.

A top-down approach will make a design easier to handle regardless of the tools that are used. A design team is not overwhelmed by complex designs and work can be partitioned easily. A top-down approach can easily be made self-documenting. Important benefits from a top-down approach also include reduced development time and improved product quality.

An important consideration to keep in mind is that design of a complex ASIC cannot be viewed as a pure hardware task. It is, in practice, a software task that is being performed by hardware designers. Complex ASIC-based designs can contain hundreds of thousands of lines of HDL code (Corcoran 1995). ASICs are designed at increasingly higher levels of abstraction. These include graphical architectural descriptions, "C" programming language models, HDL representations, and combinations of these.

Producing a quality design can be a difficult task. In order to produce designs, it is not sufficient for designers to simply receive training in an HDL and to then begin coding the design. Just as important are modeling techniques, style guidelines, and project level review and oversight. Planning the modeling approach in a top-down manner can prevent wasted time by designers following the wrong design path or creating a portion of the design that later turns out to be unnecessary.

A top-down approach is required to separate the definition phase of a project from the implementation phase. Higher level design elements are defined first, addressing completely how these segments of the design interact, before any attempt is made to implement further details of the design. In an approach that is not entirely top-down, lower level portions of a system are sometimes designed when a complete and polished definition of the higher levels does not exist. Design flow in a top-down design effort is from abstract to detailed. Effort is often expended in redesign when a systematic top-down methodology is not implemented.

3.1.2 Other Design Approaches.

Design strategies other than top-down are used by designers when developing ASIC designs. Two other common methods are bottom-up and progressive refinement. A bottom-up method may be employed when there is a need for reusable design with arguments being much the same as for reusable software. While it is possible to save design time with reusable parts, the potential exists for introducing problems if the reusable part does not perform precisely as the design requires.

Some designers may use a bottom-up approach when critical timing paths are necessary. Critical timing paths should be identified in the requirements and traceable down the design path. A purely bottom-up approach runs the risk of creating a design that does not reflect the design requirements.

Another approach used is referred to as "progressive refinement." This is essentially a combination of top-down and bottom-up techniques. The high-level design is completed first. Portions of the design are then incrementally synthesized and tested. Incremental synthesis is performed progressively until the design is completed.

3.2 LOGIC DESIGN PITFALLS.

Some of the design pitfalls that can affect device functionality are discussed in this section. Essentially, these are design practices that should be avoided but still manage to find their way into some IC designs. These pitfalls are based on faulty design techniques that are incorporated into digital circuit designs. A corollary with software programming practices can be made. Characteristics of poor software designs include the use of self-modifying code, excessive use of the "goto" statement, and poorly documented code.

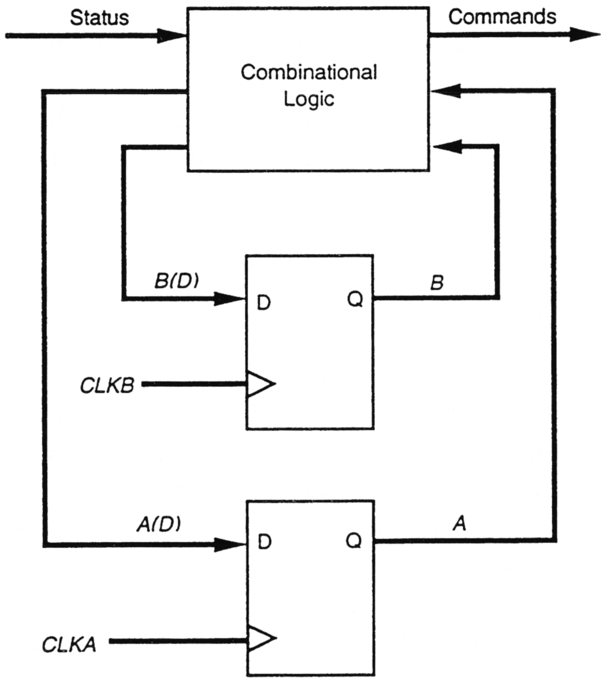
The use of tools and higher levels of design abstraction, the pressures of rapid time-to-market (TTM), inexperienced designers, and other factors often contribute to designs of inferior quality. With the availability of more powerful design tools, the actual logic implementations are further removed from the designer's critical inspection. Designs often use libraries of functions which are supplied or purchased along with the tools. Both the tools and design libraries may contain design flaws that can escape the notice of designers. Higher levels of abstraction mean that those who are not as familiar with digital design techniques and practices can now perform design functions. What can suffer is the ability of the designer to verify that the circuit implemented by the tool suite is correct. Those tasked with design verification should ensure that design pitfalls are avoided and that good design techniques are applied consistently.

3.2.1 Clock-Related Errors.

3.2.1.1 Clock Skew.

ASICs are designed using both synchronous and asynchronous logic. In synchronous designs, it is important that every clocked on-chip element receive its clock edge at the same time. Figure 11 illustrates how clock skew may occur.

Correct operation of this synchronous circuit is ensured when the *CLKA* and *CLKB* edges occur simultaneously. The combinational logic computes new values for *A(D)* and *B(D)* based on the current value of Status and the current values of *A* and *B*. A finite amount of time after the clock edge arrives, *A* and *B* will take on new values. These new values will reflect the states of the flip-flops immediately before the clock edge. The flip-flop outputs will change and the state of the combinational logic will be unsettled for a brief period. The outputs of the combinational logic will experience hazards and delays due to the finite propagation time of the individual gates of which this logic is composed. The values of *A(D)* and *B(D)* can have unstable values for a short time due to the settling of the combinational logic elements.



GSC.449.95-11

FIGURE 11. CLOCK SKEW EXAMPLE
(Winkel and Prosser 1980)

Suppose there is a skew in the clock signal so that *CLKB* is delayed with respect to *CLKA*. *CLKA* occurs and *A* takes on a new value that enters the combinational logic network. This network then requires a brief period for the logic to settle to a stable state. If, due to clock skew, *CLKB* occurs during this time, an incorrect logic value may occur at *B* causing faulty operation of this portion of the circuit. Therefore, it is important that designs be examined for potential violations due to clock skew.

3.2.1.2 Gating the Clock.

Gating a clock line will introduce clock skew and is not considered good practice. If all flip-flops attached to the clock line are not the same type (i.e., positive edge-triggered) it is necessary to invert the clock signal for some flip-flops while others connect directly to the clock. Hence, skew is introduced in the clock line. To avoid this problem, all flip-flops should be of the same type.

3.2.1.3 Clock Path Length.

The distribution of the clock line is an area of concern. In synchronous designs, there may be a large number of circuit elements that require a common clock. The elements may be located in different portions of the IC requiring the routing of the clock over significant distances. Remember that as clock rates increase, resistance-capacitance (RC) delays also increase. Since submicron propagation delays are predominantly due to RC effects, the routing of signals within an IC in order to minimize these effects are a design concern. It may be acceptable to gate the clock where buffering of the clock is necessary. Figure 12 shows how this may be implemented.

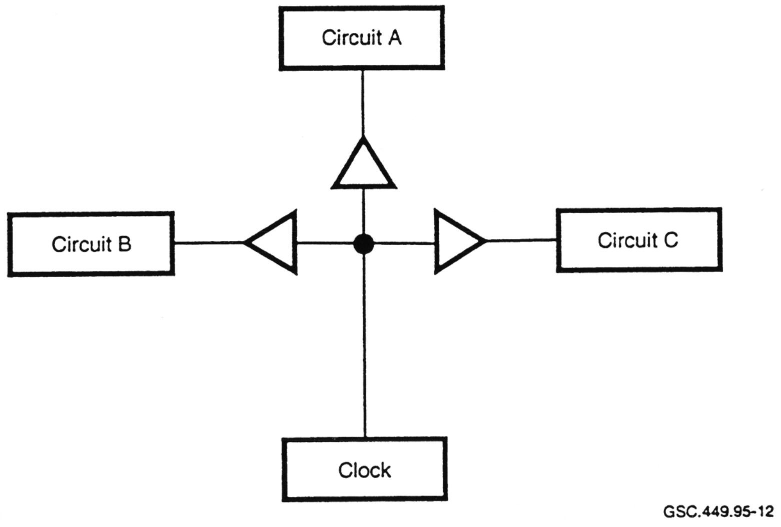


FIGURE 12. ACCEPTABLE CLOCK DISTRIBUTION TECHNIQUE
(Winkel and Prosser 1980)

These buffers should be located in physical proximity and introduce the same gate delay on each clock distribution line. The length of the set of physical lines to each subsystem and the set from the clock source to the buffers should be as physically identical as is possible.

3.2.2 Asynchronous Inputs and Race Conditions.

Many IC designs are implemented as some form of a state machine. A significant factor in the design of operationally reliable circuits is the synchronization of all input signals to the state machine. Many signals that are used by the state machine originate outside the IC and their state transitions are not synchronized with the state machine clock. These asynchronous signals can cause problems for the state machine.

Figure 13 shows a simple state machine with a single asynchronous input signal, *IN*. The individual state flip-flops require that any flip-flop input be stable during its specified setup time. If any of the flip-flop inputs change during this setup period, the flip-flop output values will be unpredictable.

If the current state is 00 and *IN* is true, then the next state will be 00. If *IN* changes to false, then the inputs to both of the state flip-flops will change to 11 so that the transition to the next state 11 can be accomplished on the next edge of the clock. However, if *IN* changes during the setup time of the

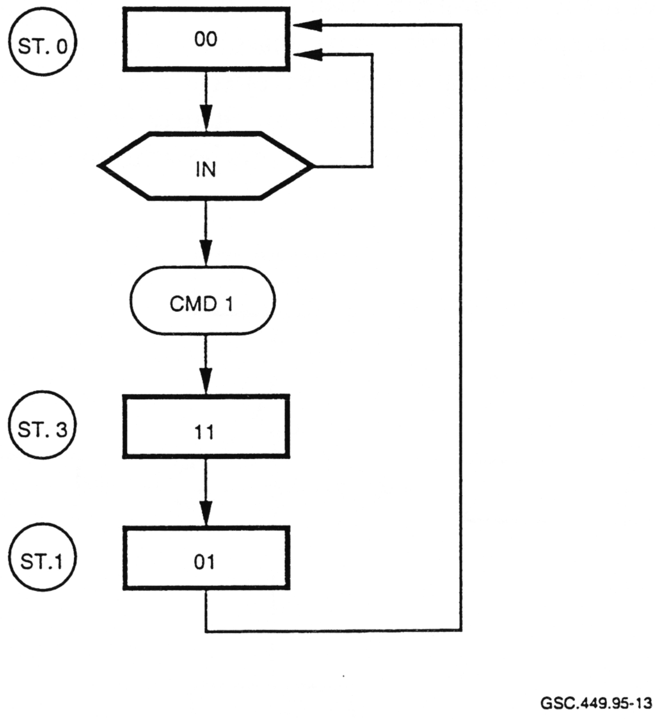


FIGURE 13. A THREE-STATE STATE MACHINE EXAMPLE
(Winkel and Prosser 1980)

flip-flops, then the next state could be 00, 01, 10, or 11. A transition to either state 01 or 10 would mean a malfunctioning state machine since the only valid next states are 00 or 11. This unsettled condition at the input is called a transition race. Digital designs should have all asynchronous inputs synchronized by the system clock to ensure that transition races do not appear.

Output races can also occur in designs where asynchronous signals occur. In figure 13, note that the conditional output *CMD1* is false when *IN* is true and true when *IN* is false. If *IN* transitions from true to false late in the state 00 time frame, then *CMD1* will be true for only a brief period of time before state 11 is entered. This causes an output race condition to occur since the output *CMD1* can vary from being one full clock cycle down to a very short pulse in duration. A very short pulse may not meet the input requirements for the circuit to which it connects resulting in faulty operation of the state machine.

While there may be ways to reduce the effects of asynchronous signals on state machines, the best way to deal with these signals is to make them synchronous. This starts at the top design level where signals are specified. Races can be eliminated from the design by using an extra flip-flop for each asynchronous signal and clocking these flip-flops using the state machine clock.

3.2.3 Asynchronous State Machines.

Asynchronous State Machines are state machines that are not dependent upon a clock, but on input changes to cause state transitions. There are a number of serious problems that can arise with this type of design, including (Winkel and Prosser 1980):

- Any noise or instability of the input signal can cause spurious state transitions.
- Asynchronous circuit theory is complex and involves numerous special cases; it may invoke restrictive design conditions.
- Asynchronous circuits can be difficult to debug.

Because of the severity of the problems that can arise with this method, it is best to avoid using this type of state machine. Synchronous state machines are much easier to debug. They can also be run with a clock frequency that varies anywhere from 0 Hz to the maximum specified clock rate. A single-step state machine with an observable state counter facilitates the debug process.

3.3 DESIGN LOGIC EXAMPLES.

This section will examine some of the design practices that can result in unreliable operation of ASIC designs. Examples of poor design techniques and some solutions will be given.

3.3.1 Asynchronous Delay Generation Logic.

Figure 14 shows the use of a string of cascade inverters for producing a pulse based on the propagation delay of these gates.

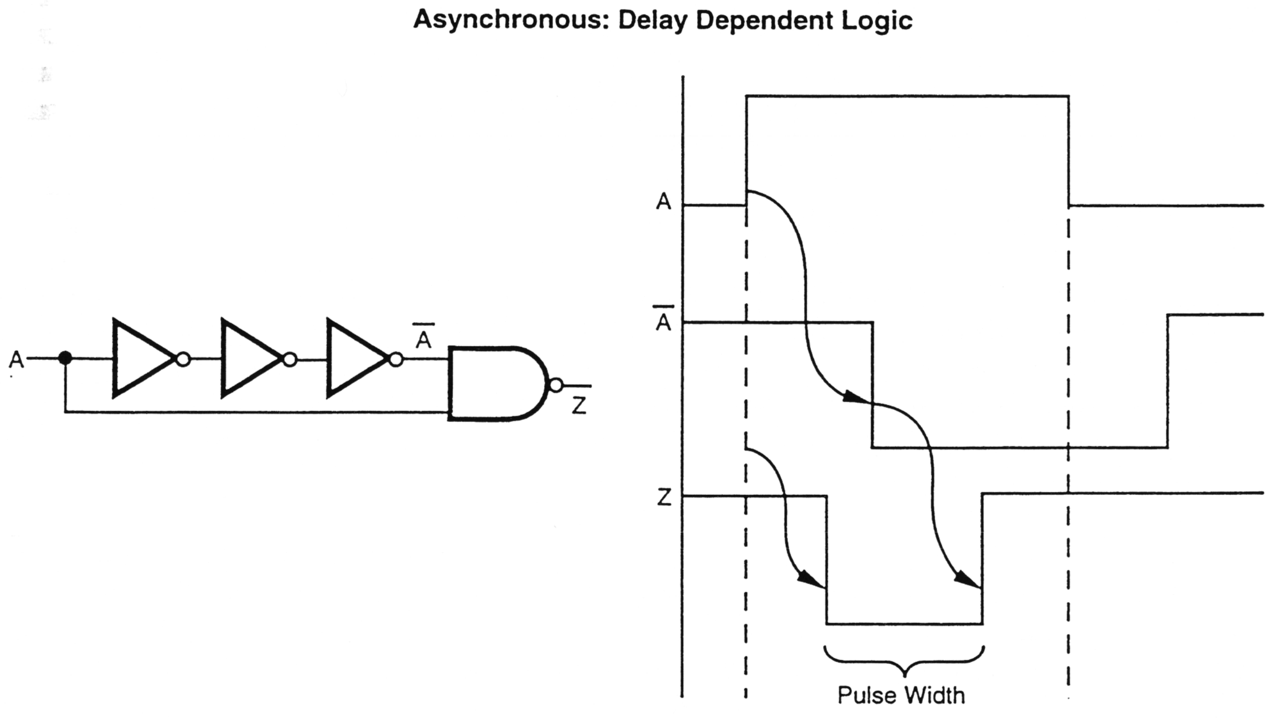


FIGURE 14. DELAY-DEPENDENT LOGIC
(Zeidman 1994)

A pulse generated by this circuit will not be reproducible under changing conditions. For instance, if the temperature changes, a different logic type is used, or variations occur in the supply voltage, the pulse width will vary. A better technique for producing delays is shown in figure 15. The delay produced by this circuit is based on a clock signal. Therefore the delay is based on the stability and repeatability of the clock signal.

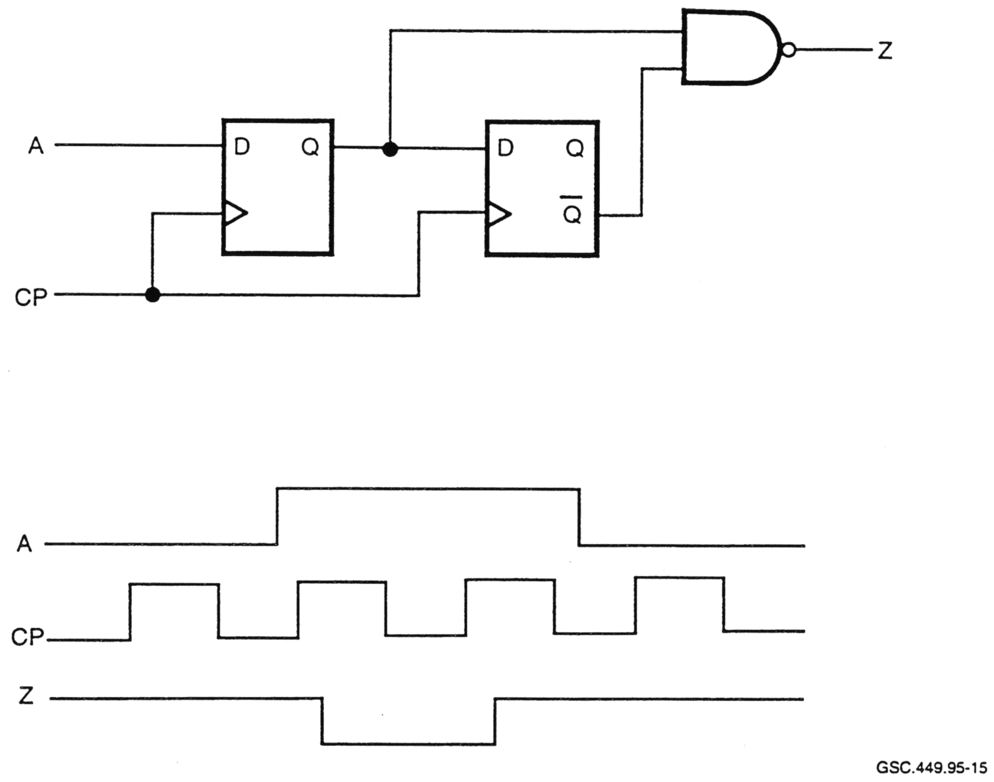


FIGURE 15. CLOCK-BASED INDEPENDENT DELAY LOGIC
(Zeidman 1994)

3.3.2 Asynchronous Logic Hold-Time Violation.

Figure 16 shows how incorrect design can lead to a hold-time violation. A hold-time violation occurs when the input to a flip-flop changes during or after the active edge of the clock. In the figure, D2 changes at the same time as D3. Since D3 is used to clock the flip-flop, a hold-time violation results. This causes instability of the signal, D4. Note also that this design contains an asynchronous element, the right hand flip-flop, which uses D3 rather than the system clock in order to clock data through the flip-flop.

Figure 17 shows how a synchronous design can eliminate the hold-time violation caused by the asynchronous clock. The main changes are the addition of a multiplexer and AND gate and making the right-hand flip-flop synchronous by clocking it with the system clock. While D2 and D3 both become active at the same time, there exists no hold-time violation since the system clock is used consistently and the signal D3 is not permitted to violate setup and hold-times at the right-hand flip-flop.

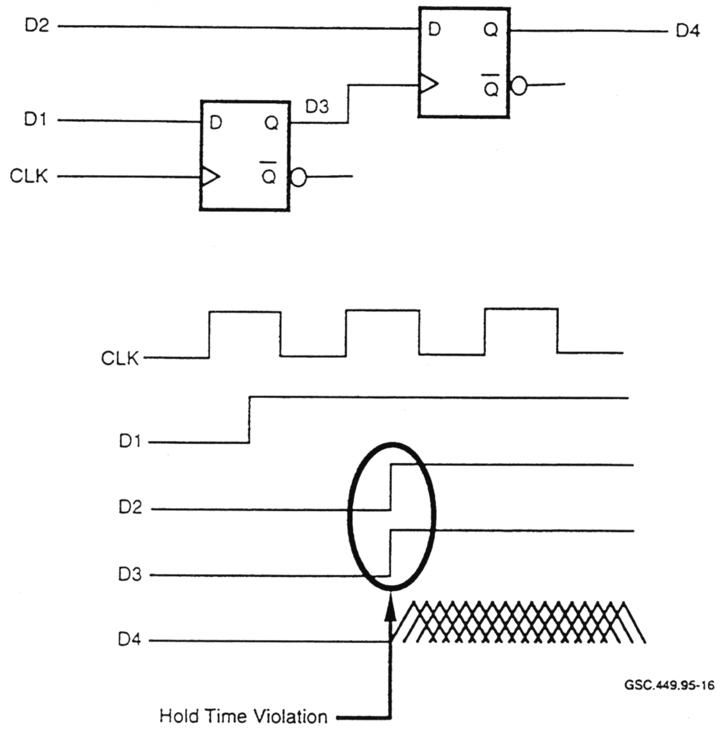


FIGURE 16. HOLD-TIME VIOLATION
(Zeidman 1994)

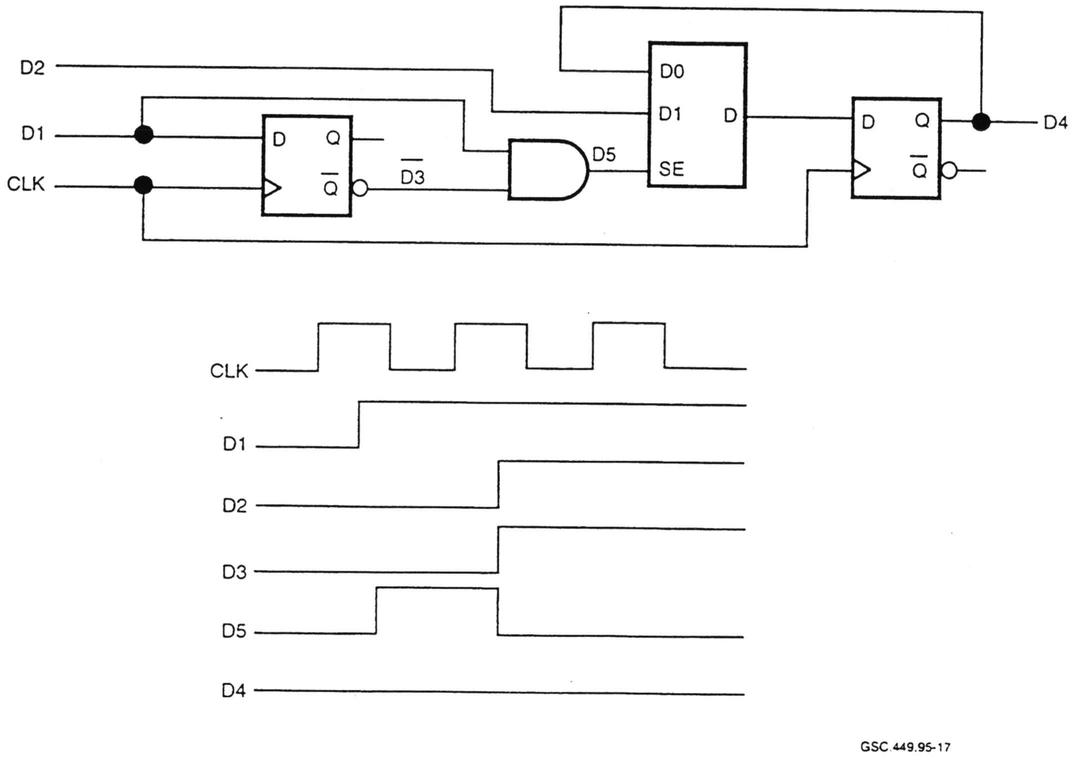


FIGURE 17. SYNCHRONOUS DESIGN WITH NO HOLD-TIME VIOLATION
(Zeidman 1994)

3.3.3 Design Glitches.

Another problem that can be eliminated with the use of synchronous design is the glitch generated during logic switching. Figure 18 illustrates how a glitch can be generated. Essentially, it is caused by the SEL signal which causes the AND gates to switch at different times due to the inverter in the signal path.

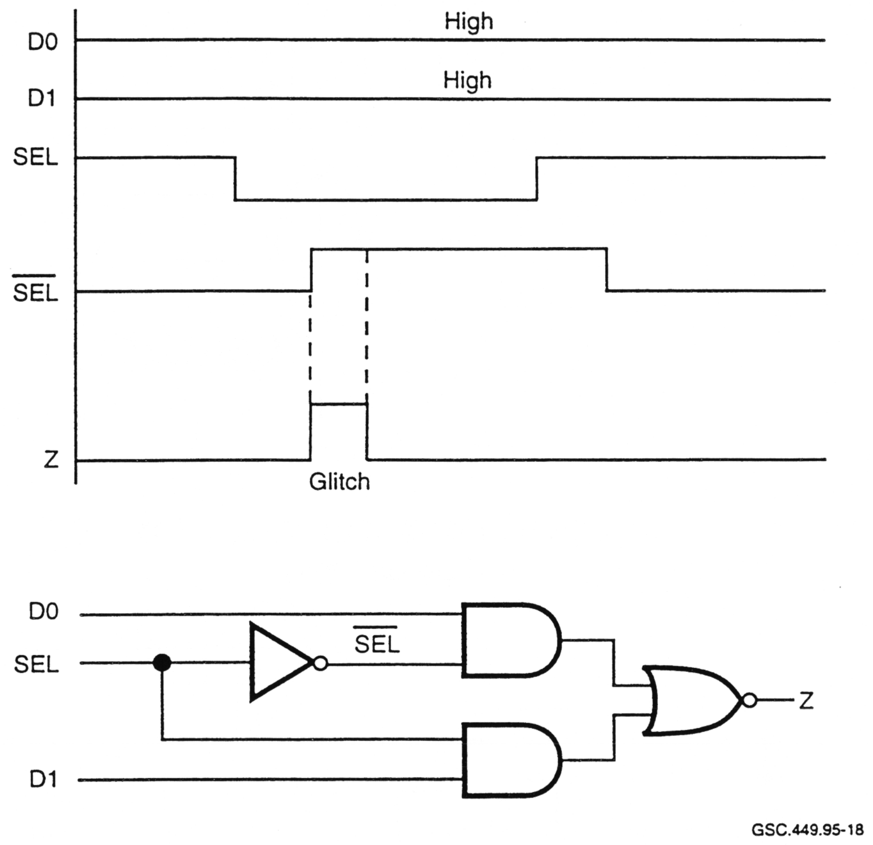


FIGURE 18. GLITCHES CAUSED BY ASYNCHRONOUS DESIGN
(Zeidman 1994)

The addition of a flip-flop to the output signal Z will make this circuit synchronous. However, the signals D0 and D1 should be synchronous with the clock in order to avoid hold-time or setup-time violations.

3.3.4 Bus Contention.

Bus contention is a potential design problem in ASICs. It usually results from a bus driver control design that is not mutually exclusive. Figure 19 shows a simple example of how this can occur.

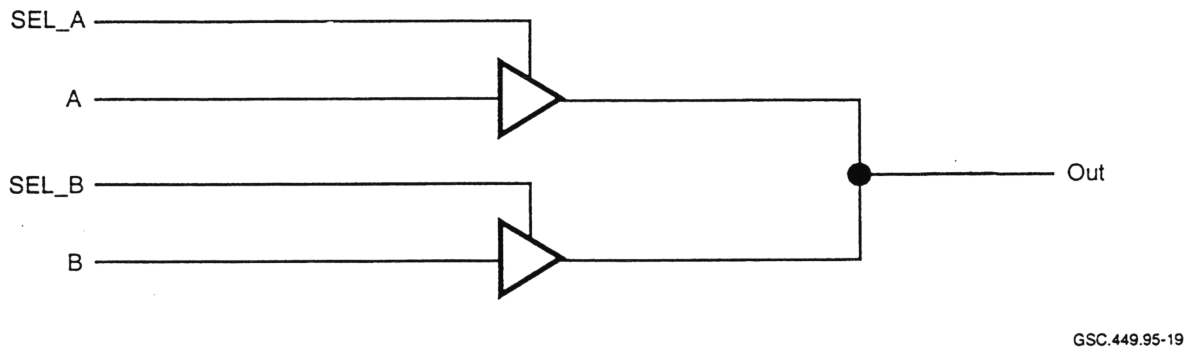


FIGURE 19. BUS CONTENTION DUE TO FAULTY CONTROL DESIGN (Zeidman 1994)

Since SEL_A and SEL_B are not mutually exclusive by design, it is possible that both drivers can be active at the same time. Bus contention can cause large current flow and damage to the IC. Figure 20 shows one possible solution for avoiding the contention problem.

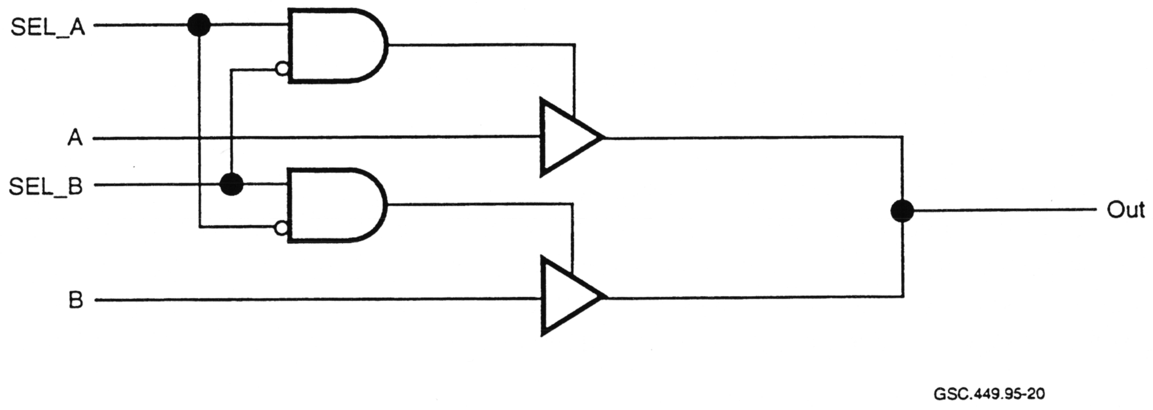


FIGURE 20. AVOIDING BUS CONTENTION BY DESIGN (Zeidman 1994)

3.4 METASTABILITY.

In many designs, situations exist where it is necessary to interface to external signals that are not synchronized to a device's internal clock. These situations can arise when signals must pass from one system to another where different system clocks are used. Also, inputs to a device that originate outside the system will not be synchronized with any internal device clock. This will occur when using a push button, for instance. This condition can result in metastability, a phenomenon that is often ignored in digital designs. Ignoring metastability, however, can lead to device failure.

A device manufacturer's specification, referred to as "setup time," must be met to ensure that metastability will not occur. Violation of the setup time required for a flip-flop is the most common cause of metastability. Data must be present and stable at the input to the flip-flop for a finite period of time before the active edge of the clock signal occurs.

In many devices, such as PALs, there are logic arrays through which the data passes, that add to the setup time. Care must be exercised by designers to ensure that the longest path taken by the data will not cause a setup-time violation. Figure 21 shows how the setup time is specified for a positive-going clock input of a flip-flop.

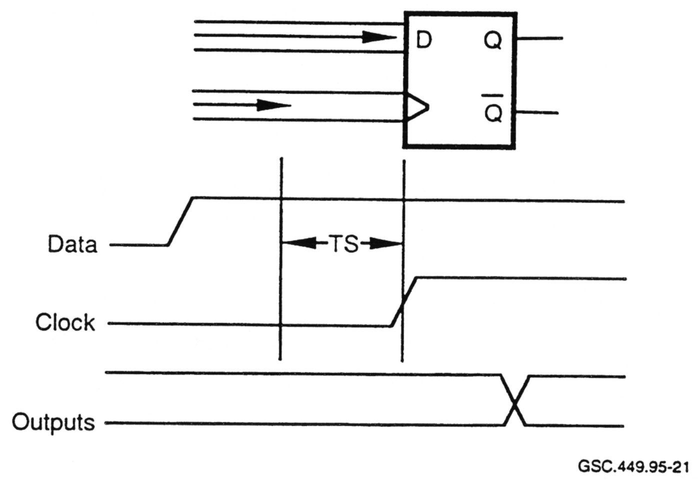


FIGURE 21. SETUP-TIME ILLUSTRATION FOR POSITIVE EDGE CLOCKED FLIP-FLOP

When the setup time is satisfied, the signal arrives at the flip-flop input well in advance of the clock. If there is a violation of the setup time, the output from the flip-flop will be unpredictable. If the relationship of the clock and varying data input are just right, the flip-flop output will be unstable for some time before it settles to a stable state. When this occurs, both the time that it takes for the flip-flop to reach a stable state and the final state that is reached are unpredictable.

Since metastability is caused by asynchronous inputs, the most obvious way to deal with this problem is to synchronize all input signals. This is done by adding a flip-flop in the path of the asynchronous input signal. In so doing, additional time is added to the signal path. If there is instability, it will be reduced in the period between clock edges. Figure 22 illustrates this signal synchronization.

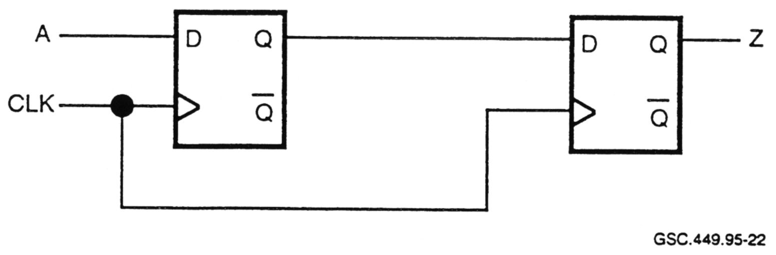


FIGURE 22. SYNCHRONIZATION OF ASYNCHRONOUS SIGNAL

One of the drawbacks of this technique is that a delay equal to the clock period is added to the signal. While this may not be a problem for an external input signal, such as a push-button event, it could be for a time-critical system signal. Also, while the synchronization of an asynchronous signal greatly reduces the occurrence of metastability, it does not guarantee that metastability is eliminated. Finally, the second flip-flop can also enter the metastable state. This can occur if the first flip-flop does not recover and reach a stable state in order to meet the required setup time of the second flip-flop.

Metastability occurs generally where the setup conditions are not met. In synchronous systems where setup time and other timing is a primary concern, metastability is not a problem since the design accounts for it. In cases where asynchronous signals cannot be eliminated, metastability will occur. What needs to be done in these cases is to ensure that the system will not fail when metastability does occur.

3.5 ISSUES IN SUBMICRON TECHNOLOGY.

Shrinking IC geometries is key to several benefits for both IC manufacturer and end user. The number of transistors can be increased yielding greater functionality per unit area. Performance is increased, since smaller geometries allow higher clock rates. Also, the ability to put greater functionality into a smaller area can yield an overall power reduction.

One manufacturer had changed a fabrication process from 0.8-micron dual-layer-metal to 0.6-micron triple-layer metal. This resulted in a 55 percent reduction in the die size and yielded a 75 percent increase in device performance. From a manufacturing standpoint, this new process allows many more devices to be created from a single die reducing overall device cost. Table 3 shows some of the devices currently available from various manufacturers and some of their important features.

TABLE 3. CURRENTLY AVAILABLE APPLICATION SPECIFIC INTEGRATED CIRCUITS (Gallant 1995)

Part Number	Gate Length	Gate Density	Power Dissipation (μW/MHz/gate)	Metal Layers
HL400C	0.5	500,000	0.8	3
CG51/CE51	0.5	754,000	1.2	3
CMOS 5S	0.36	1.6 million	N/A	6
500K Series	0.38	1.5 million	1.0	2-4
CMOS-9	0.35	2 million	0.9	2 and 3
CMOS Cell-Based Gates	0.35	5 million	> 1.0	2-5

As can be seen from table 3, the power dissipation of ASICs can vary substantially. A difference of 0.4 (1.2-0.8) μW/MHz/gate can generate a significant amount of heat in a device with one million or more gates.

Accurate estimates of design parameters such as power dissipation and delays take on greater significance as designs move into the submicron and deep submicron (0.5 micron and below) regions. Unless a strong link exists between front-end and back-end tools, accurate estimates will not be possible. Front-end tools handle tasks such as synthesis and partitioning while back-end tools handle layout and block placement issues. Without a good correspondence between these tool types, the designer of a complex ASIC is destined to perform a higher number of iterations before reaching an acceptable design. Figure 23 shows how the number of timing-related design iterations can increase as the discrepancy between the prelayout and postlayout delays increases.

Some of the key issues that designers will confront when dealing with submicron designs include:

- Accurately accounting for interconnect delay as the dominant delay factor when modeling the ASIC and design analysis based on high frequency operation of the IC.
- Designing with an accurate estimate of the power dissipation, both for potential "hot spots" and for the entire IC.
- Managing the stronger electric fields as transistor line widths decrease.
- Integrating IC package parameters into the simulation.
- Managing a complex IC design database that can exceed 1 gigabyte for current designs.

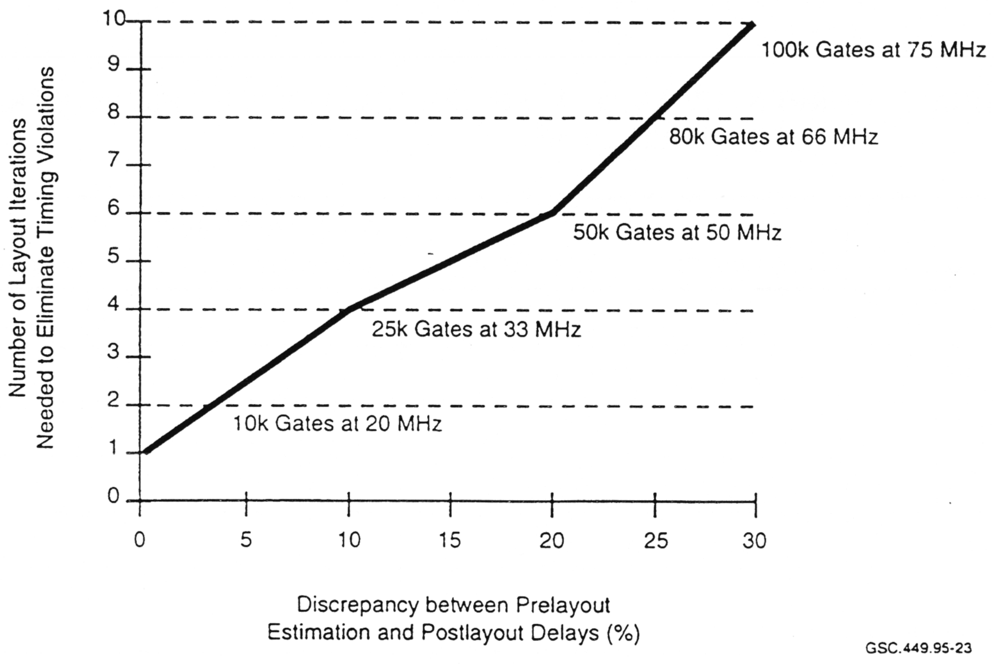


FIGURE 23. LAYOUT ITERATIONS VERSUS DELAY DISCREPANCIES (Lipman 1995)

One ASIC supplier predicts that by the year 2000 they will be supplying parts with up to 5 million usable gates. Also, they predict gate speeds to be below 100 ps and power consumption to be less than 1 μW/gate/MHz. It is anticipated that gate densities such as this will be reached using 0.15 micron technology (Waller[1] 1995). Of necessity, the operating voltage, and therefore power dissipation, will be reduced to 1.5 V. A reduction in the operating voltage also means that additional consideration should be given to these devices, since their noise immunity will be drastically reduced.

Two major objectives of ASIC designers are to implement the desired functionality and to do that within the given constraints. A device specification will include parameters such as power consumption, die size, packaging, and noise margins. HDLs are generally utilized to implement ASIC functionality. The gate-level netlist generated from the synthesis process determines the major portion of the physical design characteristics. However, unless these characteristics are strongly linked to some of the key design parameters for complex ICs, the design may be plagued by a number of potentially costly problems requiring more design iterations.

3.5.1 Gate Delay.

For shrinking device geometries, optimization of the physical layout is becoming increasingly important. Some feel that as device geometries reduce into the submicron level, physical layout optimization takes on more importance than gate optimization. This means that electronic design automation (EDA) tools now need to address the physical design constraints of submicron devices *during* the synthesis process instead of afterward. Unless these constraints are addressed, a large number of design iterations (synthesis, place and route, silicon fabrication, and test) will be performed by designers of submicron devices using conventional tools (Smith 1995). Figure 24 shows a graph of gate delays and interconnect delays plotted against various device sizes.

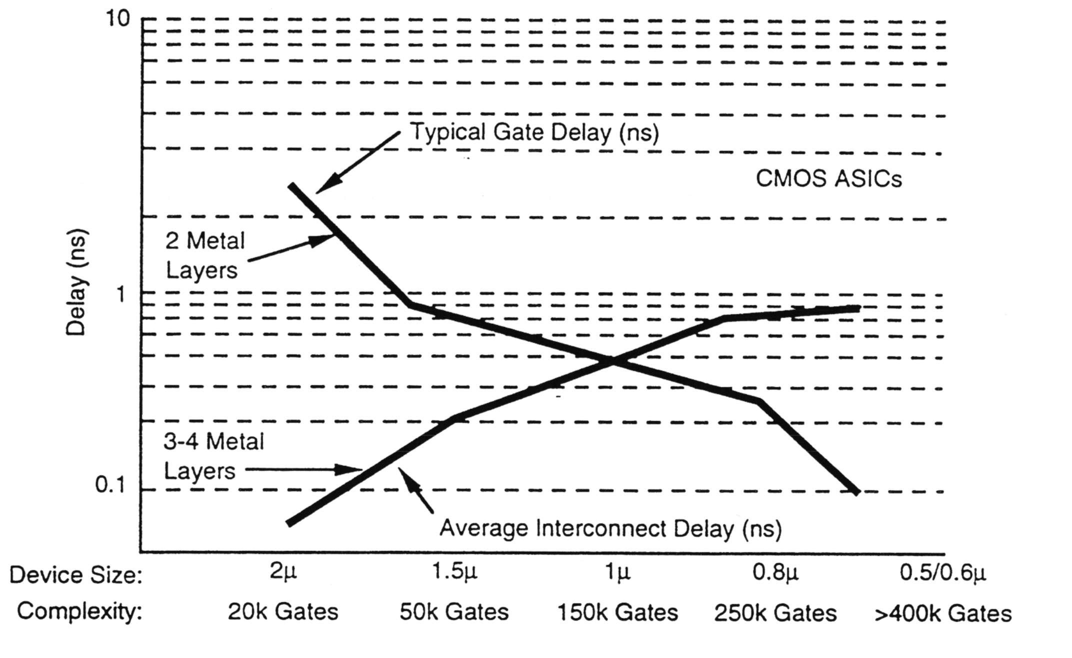


FIGURE 24. INTERCONNECT AND GATE DELAYS VERSUS DEVICE SIZE (Smith 1995)

This figure shows that even at 1 micron interconnect delays can be significant. For device sizes below that, they become more significant and exceed the gate delays at about 0.9 microns. At 0.5 micron, the interconnect delay accounts for approximately 80 percent of the overall IC delay. It appears that in the future, as device sizes decrease further, gate delays will become insignificant and EDA tools will deal primarily with interconnect delays.

The reduction in the size of the transistor has been more rapid than the reduction in the interconnect line width. As transistors shrink in size, the propagation delays they induce also decrease. This phenomenon has caused designers to use modeling techniques that account for the RC properties of the IC wiring in order to more accurately examine critical timing issues.

Having tools with the capability to estimate correctly the interconnect delays at the top design level becomes increasingly important as designs move to the deep submicron level. Interconnect resistance and interconnect capacitance both require accurate modeling. Additionally, in order to obtain more accurate models, the distributed RC equivalent circuit model must be used by the EDA tools and not the lumped equivalent circuit. The lumped capacitance model assumes that all points of the interconnect wire charge at the same rate and that all gate inputs that are tied to this interconnect wire also charge at the same rate. This is not a valid assumption and leads to inaccuracies in the model. Figure 25 gives an example of a more accurate interconnect model that will allow model simulations to predict accuracy to within 10 percent of the true device performance.

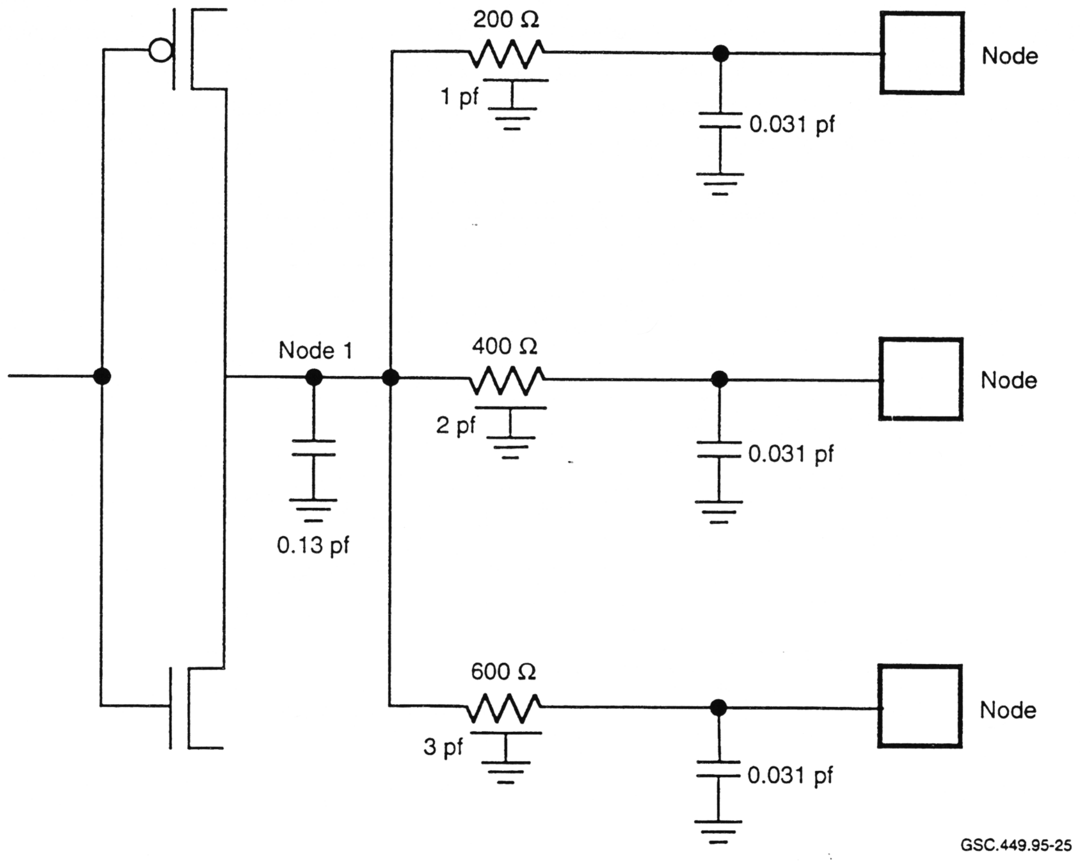


FIGURE 25. INTERCONNECT ESTIMATION FOR DEEP SUBMICRON DESIGNS (Gallant 1995)

If the designer's approach to interconnect delay is too conservative, larger line drivers are used than are necessary. This results in wasted IC area as well as excess power consumption. If the delay of critical paths is underestimated, then significant time is wasted in having to redefine portions of the circuit that did not meet the performance specification (Lipman 1995).

Reduced transistor capacitance parameters and smaller transistor geometries yield higher clock rates. This in turn can lead to other related problems. For CMOS devices, power dissipation increases with increasing clock rates. As transistor density and clock rates increase, power issues will need to be addressed on an ongoing basis.

3.5.2 Floorplanning.

In the past, floorplanning was performed as one of the final steps of the design process. The need for accurate timing information is necessary to create correct silicon designs at the first attempt. Accurate wiring delay information depends on wire length, and this, in turn, depends on the layout of the functional logic blocks and their associated I/O requirements. Floorplanner tools are used in order to facilitate this essential function.

A floorplanner is used to determine the relative positioning of an IC's major function blocks, without the need for going through time consuming and meticulous placement and routing exercises. These data are then used to make estimates on interconnect lengths and their associated resistance and capacitance values. Front-end tools use the information for generating more accurate estimates than are obtainable by statistical computation of the interconnect lengths.

Tools should provide users with effective means to reduce critical path delays and to organize and place large blocks of logic to minimize interconnects. Once a particular arrangement of blocks is made, data from this configuration can then be back annotated to the front-end tools for incorporation into a new timing analysis computation. Having greater control over the physical design allows a tighter loop among synthesis, floorplanning, simulation, and timing analysis resulting in shorter place-and-route cycles and therefore also design iterations.

Floorplanners are valuable tools for minimizing I/O delays. There are three basic approaches to minimize I/O delay using floorplanners. The first is to manually place the core component portion of the I/O macro before global placement is performed. The second method incorporates the I/O macro within a functional block. Within that block, it is placed near the corresponding I/O area. Third, an area around the core IC location is set aside for the core component portion of the I/O macrocell.

Designers often use libraries or macrocells as building blocks in designs. While accurate timing information is available for these macrocells, designers do not have the timing information for wiring between the cells since wiring delay is directly related to the final layout.

3.5.3 Crosstalk.

Designers of dense ASICs now resort to modeling to account for effects of capacitive coupling between wires. This coupling, also called crosstalk, can occur between wires on the same layer as well as between wires on adjacent layers. According to Dudzinski (1995):

> A combination of three factors influence signal integrity: signal reflection, interconnect delay, and crosstalk. Of the three, crosstalk is probably the least understood and the most difficult to detect and manage. However, its impact on circuit performance has become a first-order effect you must take into account with a routing solution.

Crosstalk is the unintended interaction of one circuit with another due to mostly capacitive, and to a lesser degree, inductive coupling. Performance is degraded by crosstalk since delays are introduced and signal purity is affected. Signal delays can cause problems for critical timing paths. Signal purity is important since undershoot and overshoot of a signal due to crosstalk can cause an incorrect logic state to propagate through a circuit. Signals that are particularly susceptible to crosstalk degradation include clock and reset lines.

Noise coupling into and out of nets on an IC also requires attention from designers along with adequate tools to identify and resolve integrity issues. Noise coupling between nets is the total noise impinging on receiving nets from the surrounding transmitting nets. Each net on an IC can have different transmitting and receiving characteristics. When the net's maximum accumulated noise exceeds the maximum specified level, signal integrity for that net is jeopardized (Dudzinski 1995).

Since crosstalk is due largely to capacitive coupling, there are several ways that designers can minimize problems caused by crosstalk. Long parallel lines should be avoided. This includes lines that are on adjacent layers. Also, as signals are further removed from the voltage plane, crosstalk becomes more of a problem. EDA tools should be used to alert designers to potentially troublesome layouts such as long parallel lines.

If the crosstalk problem is not addressed during design, then it may surface during device test. Another iteration at this point is costly and also causes schedule delays. If marginal devices slip through device test and are assembled into products, unreliable products can result.

3.5.4 Power and Thermal Design Issues.

As device complexities have increased, one portion of the design phase has taken on increasing importance. This is the heat management and accurate power estimation. The majority of current designs are implemented using CMOS technology. Factors which cause power consumption in these circuits result from the following (Lipman 1995):

- Switching power—generated due to the charging and discharging of interconnects and transistor gates when logic signals change states. Switching power typically consumes from 70 to 90 percent of the total device power.

- Dynamic short-circuit power—a CMOS circuit phenomenon where the output pull-up and pull-down resistors temporarily cause a short of V_{cc} to ground during a change of state of the output node. Typically 10 to 30 percent of the total power of a device is consumed by dynamic short-circuit power.

- Direct Current (dc) leakage—occurs when the CMOS transistors are not fully on or off. Generally less than 1 percent of the total power is consumed by dc leakage.

Power dissipation is a measure of the heat that a device generates. Using 0.35 micron technology, it is possible to make a device with 4 megabits of memory and 2 million gates. Power dissipation figures of 0.8 μW/MHz/gate and 1.0 μW/MHz/gate are commonly used. Table 4 gives the power dissipation required for two different devices.

TABLE 4. POWER DISSIPATION EXAMPLES

Gate Power (μW/gate/MHz)	Gates	Clock (MHz)	Maximum Power Dissipation (Watts)
1.0	500,000	100	50
0.8	2,000,000	175	280

A consequence of decreasing device line widths is that the number of gates per device and the device clocking frequency both increase. Actual power consumed will be less since not all gates of a device toggle at the same time. However, it is easy to see that power dissipation problems will continue to grow, along with device complexity. Device layout and packing issues are becoming increasingly important for device reliability. In addition to reliability issues, system cost and size are influenced by power dissipation issues.

When facing these issues, having the assistance of the right tools can mean the difference between an unreliable and a reliable design. The tool should also account for and integrate the electrical and thermal characteristics of the IC package. Tools will allow designers to predict not only total power dissipation for the IC but also power dissipation for individual blocks of logic. Knowing the total power dissipation allows the designer to choose the correct IC package and also to design an adequate box containing the IC and associated printed circuit board. Individual logic blocks can be reworked by modifying the clocking system or other block parameters.

Many ICs are designed for low power operation. This is true for portable systems and others where reducing power can result in significant cost reductions. EDA tools that address power analysis will become more commonplace as levels of on-chip integration increase.

The manner in which power is distributed, normally not thought of as a design issue for ICs, has become an issue of increasing concern as transistor density and clock rates increase. As with crosstalk, tools can be of great assistance to designers when dealing with power issues. Power-related issues can be dealt with by designers in a number of ways. Following are some techniques that designers use to mitigate problems caused by excessive power dissipation:

- Reduce the power supply voltage. A three-volt power supply can reduce power by more than 50 percent.

- Minimize clock distribution trace lengths.

- When possible, use a low voltage I/O interface.

- Use high-speed serial data transfer instead of wide bus data transfers where possible.

- Use heat sinks/fans on problem ICs.

- Identify areas of high current usage with available tools. The designer can investigate alternative methods to accomplish the same goals at reduced power.

- Add built-in power-down circuitry so that all or a portion of an IC can be switched off.

- Eliminate power-robbing glitches from the design.

Tools can provide designers with estimates of the device's current distribution and power dissipation. This information is useful in the design of ground buses and distribution networks. If there are heat sensitive areas on an IC, such as analog function blocks, tool estimates can provide data that designers need to produce circuits that operate more reliably.

In order to address the new issues of submicron technology and reduce the number of design iterations, three key elements need to be in place. They are (Smith 1995)

- constraint-driven place-and-route tools,
- physical design constraints applied to the tools, and
- synthesis tools that account for route planning.

Conventional design methods apply top-level constraints manually to EDA tools. Increasingly dense ICs cause this method to fail, requiring numerous design iterations until an acceptable solution is reached. It is possible that top-level constraints can be translated to gate-level constraints by using a constraint-driven place-and-route methodology. This, however, requires that new EDA tools address these issues at the top level. Tools that consider physical design constraints need to take power, propagation delays, crosstalk, and other physical design parameters into account.

3.6 NOISE AND GROUND BOUNCE.

Signal integrity within an ASIC is a factor of which CEs need to be aware. Predicting signal integrity can be a difficult task for designers. Complicating the matter is the fact that there is a lack of tools that take signal integrity issues into account during the IC layout phase. Often the tools, technology, and methods are not identified sufficiently early in the total process. The device packaging technology is often put off until the final design stages; while the semiconductor technology is normally chosen during the early design stages. When this is done, two problems can result: inability to predict performance and inability to meet performance expectations.

In order to ensure design integrity, it is necessary to chose packaging and interconnect technology together with

- selection of semiconductor technology,
- behavioral simulation, and
- system integration plans.

Noise issues are essentially the same for IC designers as for printed circuit board designers. However, the problem complexity increases along with IC complexity. Since signals are closer together with each technology advancement, RC-based effects increase due to closer signal wires and the increasing resistance of smaller signal wires. One of the complicating factors is that those who are skilled in digital technology are not skilled in analog techniques. As device complexities and clock rates increase, signal integrity becomes more of an issue.

Some of the fundamental principles found in interconnect technology have similar bearing on IC design. These principles include (Merkelo and Liaw 1995):

- Not only the path quality but also the return path quality of the signal are significant factors for signal integrity.

- A flow of signal current causes an equal and opposite return current flow. An impedance change influences both currents.

- Changes in geometries yield electromagnetic discontinuities. This, in turn, causes a concomitant reflection of the voltage and current.

- A digital signal is not influenced by termination conditions until it propagates to the termination point.

- The existence of another signal or other metal in proximity increases the likelihood that there will be capacitive and inductive interaction.

- The signal interaction is data dependent.

- Severe discontinuities are created when multiple interconnections are made. These discontinuities affect both active and inactive circuits. The number of quiescent circuits and the number of circuits that are switching simultaneously proportionally influence signal quality.

3.6.1 On-Chip Propagation Characteristics.

Low voltage operation, high complexity, and high speed are favored by miniaturization. However, two major drawbacks that must be overcome by miniaturization are high values of wire resistance and thin dielectric fabrication. Both are critical issues for miniaturization since both can impact signal integrity severely.

ASICs and memory ICs are fabricated using multiple layers of metal for interconnections between the various circuit elements. Memory ICs have regular patterns and require from one to three layers of metalization. Complex ASICs (which may also contain memory) generally have much more random patterns and can require many more layers of metalization in order to implement the interconnection of circuit elements. While the technology exists to fabricate many levels of metalization, there are other issues that need to be addressed during the design, such as signal integrity, signal synchronization, and fabrication cost.

Another difference between old and new IC technology is that on-chip signal propagation loss, which once was not a factor, must now be accounted for in complex designs. Formerly, signal propagation did not need to account for line propagation delay since lines were physically larger (lower resistance) and gate delays were also much larger. Delay is now RC-dominated. Since line geometries have become small in order to support the massive amounts of on-chip circuit elements, line resistance has increased enough to be the dominating delay factor for on-chip signals.

Ideally, low-loss lines are required that do not take up much of the available area. However, this can be difficult to implement since as signal speeds increase, "skin effect" also increases. Skin effect causes a high speed signal to travel near the surfaces of the conductor without taking advantage of the entire cross section of the conductor. As a result, the ac resistance will be higher in value than the dc line resistance. In order for the designer to be able to model critical paths accurately, it is essential that the propagation properties be modeled accurately.

When simulation is used to model on-chip interconnections in a complex IC with small geometries, a number of factors that influence signal propagation must be taken into account. Some of these effects include (Merkelo and Liaw 1995)

- propagation characteristics,
- signal damping,
- reflections,
- fan-in and fan-out, and
- discontinuities (discrete and distributed).

When simulation is used to model signal paths, CEs need to be aware of device geometries to ascertain if the modeling accounted for all contributing factors.

3.6.2 Consequences of Noise.

Timing problems due to signal skew can cause failure of the IC. This failure can take the form of a hard logic fault and cause problems such as bus contention on read and write cycles. Logic faults generally are rare in designs that are simple. Other observations that can be made about on-chip logic faults include

- Faults are rare from crosstalk (coupling due to adjacent signals) alone. They are usually the result of a combination of factors.

- Faults are common in systems that have a large number of simultaneous switching events.

- Faults are common where inaccurate timing estimates are made. These estimates are a reflection of the simulation tool fidelity.

- Faults generally are nonlinear and data dependent. Based on geometries, the cumulative effect of simultaneous switching of certain signals contributes to these faults.

While timing-related faults are rare in simpler circuits, they increase in probability if all factors contributing to timing delays are not characterized well or if they are not characterized for all combinations of logic states. Since signal edges are the measuring rod of circuit timing, anything that influences these edges must be taken into account. Signal edges are affected by crosstalk and signal reflections.

Noise is generated when there is an incremental demand for current from a switching device, such as a transistor, and is referred to as ΔI noise. When multiple devices switch simultaneously, the incremental demand multiplies. Inductance in signal paths generates L(di/dt) voltage. ΔI noise can also affect signal edges even if the noise is not severe.

Contributions to signal edge degradation can be insidious since they are data dependent. These contributions can vary based on the state of the logic or the polarity of the logic state change. In propagation where delay is characterized by RC effects, how the capacitances are modeled can influence the accuracy of timing predictions.

When there are two parallel conductors, it is useful to represent the mutual capacitance in two parts. One capacitance being dependent upon the voltage of the surrounding conductor or conductors; the other one being relatively constant with respect to voltage. The mutual capacitance of any segment is given by

$$C_{mi} = \Sigma C_{ik}(v)$$

where i varies from 1 to k, and k is the number of neighboring leads. The mutual capacitance, which is voltage-dependent, can therefore be determined by (Merkelo and Liaw 1995):

$$C_{ik}(v) = \begin{cases} C_{ik} & V_k = 0 \\ 0 & V_k = V \\ 2C_{ik} & V_k = -V \end{cases}$$

It can be seen that as the logic states between two lines change, there are three different capacitance values that need to be accounted for when making timing predictions. When the voltage of the neighboring lead is 0, the mutual capacitance is C_{ik}. If the voltages are the same, there is no contribution of mutual capacitance. When the voltage of the neighboring lead is negative with respect to the signal lead, the mutual capacitance is doubled.

Without taking these characteristics into account, tools will not be able to produce accurate timing models. Tools that can be affected include ones that model clock distribution, timing synchronization, and propagation delays. Accurate modeling of these parameters becomes increasingly important as device complexity increases.

3.6.3 Recommendations for On-Chip Design.

High speed and signal quality can suffer when the design of the current return path receives insufficient attention from IC designers. Some of the problems that are experienced with current return path design include:

- path resistance is too high,
- return path is too long,
- return path exhibits semiconductive behavior, and
- return path is shared by too many interconnections and other circuits.

Many other concerns need to be addressed by designers, especially since IC technology has progressed to below the submicron level. These concerns include:

- Characteristic impedances of signal lines vary over a wide range basically due to incorrect signal return path design. Often signal lines do not have return paths in proximity.

- IC metalization is subject to complex coupling problems, as previously discussed.

- Mixed logic configurations, such as on-chip 5 V and 3 V logic, are more prone to hard logic errors than would be a single voltage IC.

- Effects of noise, whether internally or externally generated, should be diminished by using a meshed ground plane of proper proportions.

- Care needs to be exercised in the choice of ground pin position and quantity in order to maintain uniform characteristic impedances.

- Complex on-chip signal interactions need to be considered when making delay estimations.

- Data dependent behavior becomes a higher risk where multiple parallel signal coupling and multiple return paths are present.

- Minimization of interconnect wires is highly desirable, especially for signal integrity. Design partitioning, often performed during logic synthesis, should be given careful consideration in complex designs.

- On-chip global interconnections should be kept as short as possible. Inverters that are reasonably placed can reduce noise coupling in the lines.

Unless noise truly originates from an off-board source, it is a design problem. Noise can originate from within the IC or can propagate from printed circuit board traces to the IC pins and into the IC. Being data-sensitive, it may not be recognized as a problem until devices suffer field failures. It is more a problem in complex and high speed ICs, such as ASICs. CEs need to be aware of these issues and check to see how manufacturers have addressed them.

3.7 LATCH-UP.

Fabrication of CMOS ICs involves silicon processes that create parasitic silicon-controlled rectifier (SCR) circuit structures. While not intended to operate in the SCR mode, these structures, when subjected to certain conditions, exhibit SCR characteristics. Figure 26 shows the basic structure of the parasitic SCR.

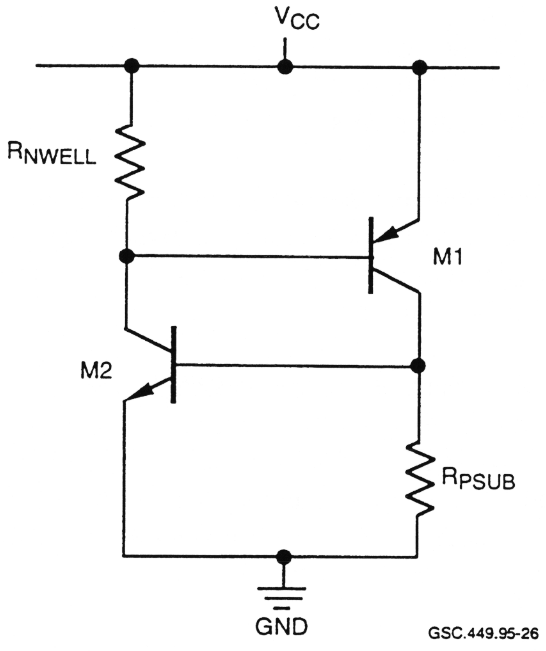


FIGURE 26. PARASITIC SILICON CONTROLLED RECTIFIER STRUCTURE

In this circuit, complementary NPN and PNP transistors are cross-coupled having common base-collector regions. The base area of the PNP transistor is composed of N-well diffusion while the emitter is formed from P-type source-drain regions and the collector from substrate regions. The NPN transistor has a base formed from the P substrate, emitter from the N-type source-drain, and collector junction from the N-well diffusion.

Normally, only "leakage" current is flowing in the SCR structure and the SCR is in the blocking state. When a current develops across any of the parasitic resistors, a voltage drop is generated which produces a forward bias across the parasitic base-emitter junction. This bias allows a collector current flow in the transistor. The collector current that flows across the base-emitter resistor generates a voltage sufficient to cause the transistor to conduct. When this condition develops in one of the transistors, it produces a forward bias across the other transistor, which now also begins to conduct. This creates a regenerative condition where the currents flowing become self-sustaining.

As is the case with an SCR circuit, once the current starts flowing, it cannot be stopped without interrupting the emitter-collector current path. Hence, even though the source that originally caused the current flow to begin may have been removed, the current flow continues. In the SCR configuration, the transistors operate as low-resistance high current switches. Once this occurs, this latch-up current flow can cause permanent damage to this circuit, to the transistor junctions, or to the metal lines that connect the circuit elements.

Latch-up can occur in several different ways. Among the most common are

- power supply overvoltage,
- overshoot/undershoot of I/O pins,
- improper application of power to device, and
- problems with the fabrication process.

Latch-up is one of the potential problems that can be caused by power supply overvoltage. Supply voltage that exceeds the device's rating can cause the breakdown of internal junctions. Proper design of the power supply, connecting systems, and power distribution is necessary to minimize the likelihood of overvoltage-induced failures.

I/O pins can be very noisy as a result of being connected to printed circuit board traces that are subject to capacitively coupled signals from other traces. Overshoot of a signal can occur when a fast switching signal is driving a capacitive load such as the printed circuit board trace. A transient forward bias condition at the I/O transistor junction can result. Latch-up is likely to be induced under these circumstances. The occurrence of latch-up can be reduced using care in the design and layout of the printed circuit board and attached devices.

How the IC is powered-up can also cause problems. If device pins are driven before power is applied to the IC, latch-up may result. This can occur if the IC is plugged into a powered socket or if a printed circuit board is plugged into a powered backplane connector. When this happens, the input diodes may become forward biased and coupled with the delayed application of power to the IC may cause latch-up. Designs should ensure the application of device power before voltages are applied to the I/O pins, and designers should ascertain from manufacturers whether or not their devices have any designed immunity to latch-up.

Latch-up is also viewed as a failure mechanism in IC reliability research. It can result from improper design or fabrication techniques. Quality control in the fabrication process is essential for the reduction and elimination of latch-up problems.

3.8 SINGLE EVENT UPSET OF DIGITAL LOGIC.

The upset of digital systems has long been recognized as a possibility for satellite and other space-based systems. Older IC technologies are unlikely to be influenced by single event upset (SEU). Until recently, it was not considered a problem for commercial avionics. However, as submicron technology advances continue, experts believe that SEU will become more and more a problem. According to Keller (1993):

> Systems integrators, avionics manufacturers, and even some of the major commercial aircraft manufacturers are finding that cosmic radiation poses an SEU threat to avionics flying at altitudes exceeding 10,000 to 15,000 ft.

The most vulnerable devices as far as susceptibility to SEU are SRAMs. This is due to the way SRAMs are fabricated. The individual transistors that make up the memory cells are tightly packed. Each new generation of memories further reduces the spacing between transistors, increasing the probability of a particle hit. As memory dimensions continue to decrease, it is possible that upsets could occur as often as once per flight unless precautions are taken.

While ASICs are widely used in modern aircraft, such as the Boeing 777, they have not seen acceptance in space applications. This is due in part to uncertainty about the SEU hardness of ASICs.

In a study conducted by Boeing, it was found that SEU rates increased by a factor of 2.2 between mid and high altitudes, and that an additional increase of the SEU rate by 2.1 occurred when going from mid to high latitude. It was observed that the SEU rate over Norway is close to those found in low earth orbit (Keller 1993).

Protection from SEU is a design issue. Memories incorporated into ASICs and FPGAs may be upset by cosmic radiation. Additionally, other logic can be upset. Radiation can induce transient upset into combinatorial circuits. If the upset persists long enough to meet certain minimum register requirements, such as pulse height, width, and timing, the error becomes “switched in” to produce a stable bit error (Baze et al. 1993). Techniques that have proven effective in minimizing the effects of SEU include:

- Using radiation hardened components which are heavier, larger, and more expensive. Not all vendors offer this packaging for their devices.

- Using error detection or Error Detection and Correction (EDAC) techniques such as parity or Hamming codes.

- Changing the basic SRAM design to a higher current model.

- Shielding the memory components.

3.9 CONSIDERATIONS FOR PRINTED CIRCUIT BOARDS.

Signal integrity is a problem not only on-chip, but off-chip as well. One of the reasons why noise is such a problem is that there are impedance discontinuities along the path of a signal. As a signal travels from one IC to another it meets discontinuities at the following locations:

- originating wire bonding pad,
- originating IC lead,
- originating lead contact at printed circuit board,
- signal intersections and via holes,
- terminating lead contact at printed circuit board,
- terminating IC lead, and
- terminating wire bonding pad.

These discontinuities cause signal reflections that result in ringing and crosstalk. Noise that originates from within the IC is easily conducted off the IC. Noise generated from outside the IC can be conducted through the traces, socket, and device pins into the IC.

Crosstalk can also be an issue of concern for printed circuit board designers. The same effects that occur on-chip are also present off-chip. Long parallel traces and fast switching signals are two of the main contributors to printed circuit board crosstalk.

3.10 PERFORMANCE MEASUREMENT.

Device performance is often unknown until the ASIC is programmed and exercised in the target system. Accurate estimates on performance can be difficult to make. With FPGAs, for instance, it is necessary to "implement the design before knowing what the final performance will be" (Kapusta 1995). The placement and routing of a design determine the propagation delay and maximum operating frequency. Timing predictions for FPGAs are therefore very difficult and need to be finalized in the implementation.

Generally, timing is easier to predict when the devices are smaller. PLD timing prediction relies upon the manufacturer's data book specification. Timing for CPLDs is generally more difficult to predict than other smaller PLDs. CPLD timing can be dependent upon factors such as the number of product terms or the fan-out. The implementation decisions are typically done by a software logic compiler. Therefore the performance may be unknown until actual device testing can be performed. Some CPLDs have simpler timing specifications, allowing the designer to predict accurately beforehand the expected performance.

4. TOOLS AND TECHNIQUES FOR HARDWARE INTEGRATED CIRCUIT DESIGN.

4.1 EARLY DESIGN TOOLS.

The first significant tools that became available to assist IC design were computer aided design (CAD) tools. CAD tools were designed to assist in the layout of printed circuit boards. The printed circuit board was a target for greater IC integration by reducing the trace widths, spacing, via holes, and distance between the ICs.

As digital technology continued to evolve, other computer-based design tools emerged. These included schematic capture, netlist generators, and fan-out checking tools. When PLDs became available, new design methods became essential. While manually designating dots for each interconnect of a PLD is sufficient for smaller PLDs, it becomes tedious and error-prone for more complex PLDs. The current PLD complexity level is too great for traditional techniques. Other factors such as development cost and TTM are also driving the requirement for tools that automate IC design.

4.2 APPLICATION SPECIFIC INTEGRATED CIRCUIT TOOLS.

Many of the advances in the digital circuit design field are related to the needs of ASIC designers. Logic synthesis tools and testing tools have several generations of improvement as the *average* size of design implementations for ASICs has increased to over 50,000 gates. Design platforms on which ASICs and other user-programmable device design tools run have evolved from the minicomputer and mainframe to the commonplace workstation. Tools are also now commonly found running on the high-end PCs which rival the performance of low-to-midrange workstations.

Along with the need to keep tools in step with increasingly complex very large-scale integration (VLSI) technology came changes in the way designs were captured and translated into a digital format. One of the essential steps in the technology progression was to move to higher levels of abstraction in the representation and design of digital circuits. This brought about a new category in digital design representation: behavioral models. Instead of drawing the actual logic gates, registers, counter, and other circuits, the behavior of those circuits is described in an HDL. The actual implementation is then left to the tool suite. Once a particular language is chosen, a logic synthesis tool is used to take the high-level circuit description and translate it into a netlist. This netlist describes the circuit elements and their connectivity. The netlist contains the same information that would have been produced, or deduced, manually from a schematic drawing.

The synthesis tool takes as input some form of HDL. Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) and Verilog HDL are currently the two most popular HDLs; although there are a number of others. Being the two most used HDLs, they also benefit by having the greatest amount of support from tool vendors who create tools to be compatible with these HDLs.

With the rapidly increasing gate densities, the designer's productivity is struggling to maintain pace. EDA vendors have responded to this problem by offering a number of tools that are aimed at solving a small portion of the ASIC design task efficiently. According to Beechler and Hanz, (1994):

> Today the gate array design tool suite is composed of a "best of breed" approach, in which companies piece together the best "point tools" for each phase of the design process.

Associated cost for ASIC hardware development includes the tool suite, computer hardware, training, tool familiarity, and CAD support personnel. Often, developers purchase, create, and reuse libraries of HDL-based designs. When the design is finished, the netlist for the ASIC is sent to an ASIC vendor (foundry) for placement and routing. The vendor fabricates the prototypes and returns them to the developer for testing.

User-programmable ICs, such as PALs and HCPLDs, can relieve some of the workload, time, and expense related to the ASIC development cycle. Where the placement and routing phase is typically done at the foundry for ASICs, it is done at the designer's office for PALs and HCPLDs. A computer file is then used to program the IC using one of the numerous device programming tools available. Rapid iterations are possible since these devices are user-programmable. Simple changes can be made and new devices produced in a matter of minutes.

With the pressure building steadily to improve software tools, silicon suppliers are strongly emphasizing open systems that make best use of third party EDA houses. Some vendors allow designers to mix and match tools from third party vendors. Design costs for ASICs are growing along with the complexity. Design houses need to maintain a closer relationship with the customer in order to maintain the expected fast turnaround times (Waller[1] 1995).

4.3 DESCRIBING THE DESIGN.

While dramatic improvements have been made in tool capability, describing a complex ASIC is a time consuming task that can require a large team of designers. Design tools have not evolved sufficiently to produce designs based on a thorough specification. Detailed design descriptions must be facilitated by some other means. Common methods that designers use for describing logic designs include:

- hand drawn schematic,
- state machine,
- waveform,
- logic description tools such as PALASM, CUPL, and ABEL, and
- HDLs such as VHDL, Verilog HDL, and others.

The following sections describe some of these techniques in more detail.

4.3.1 Hand-Drawn Logic.

Even smaller designs can benefit from the use of design tools. Until recently the hand-drawn logic diagram was commonly used for smaller designs. Several significant problems need to be addressed for logic designers using the hand-drawn method in today's environment.

While the hand-drawn method will work well for designs with low gate counts, it becomes increasingly difficult to use when design complexity increases. It is commonly agreed that when a design's gate count exceeds about 10,000 the design should be automated. A design rapidly becomes hard to manage and costly with increasing gate count.

The entire process becomes cumbersome with increasing complexity. The design is simply the first step in the process. It is also essential that there is a sufficient level of inspection and testing to verify design correctness. Design verification involves (minimally) a labor intensive inspection by an independent party.

Since the process is not automated, the iterations of design-test-modify become quite cumbersome and time consuming. Changes in complex designs are costly when automation is not used. Since cost is a bottom-line factor, changes late in the design-test cycle make the hand-drawn logic method a money loser.

4.3.2 State Machine.

In general, a state machine represents a process execution. It has knowledge of the current state. This current state, along with the values of select variables, determine which will be the subsequent state.

For digital hardware, the state machine has been a popular technique of design description. The algorithmic state machine (ASM) is a method used for synchronous system notation. The ASM implements control algorithms and is expressed in a flow chart language that is similar to a conventional software flow chart. Essentially, the ASM expresses the concept of a sequence of time intervals in a precise manner, where the software flow chart describes only the sequence of events, and not the time durations (Winkel and Prosser 1980).

The state machine design is commonly used for applications requiring high-speed controllers, such as bus interfaces. It is also used for modeling protocols and complex logic control functions. Also, the state machine concept is integrated into high-level design tools. Some HDLs include syntax that facilitates the description of state machine behavior.

In order to cope with design complexity, synthesis tools implement the finite state machine with data path (FSMD) model. The FSMD is flexible, can model any digital design, and is not limited by the number of states that need to be implemented. More on this model is found in section 4.4.4.2.

A state machine that is designed manually involves a limited number of well-defined steps. They are

- Design the control algorithm and represent it using a state diagram.
- Create the state transition tables from the state diagram.
- Translate the state transition tables into logic.

The control algorithm can be represented by one of three methods: Moore, Mealy, or Moore-Mealy. How the output is determined decides the state machine category. Figures 27 and 28 are block diagrams that demonstrate the basic operation of the Moore and Mealy state machines, respectively.

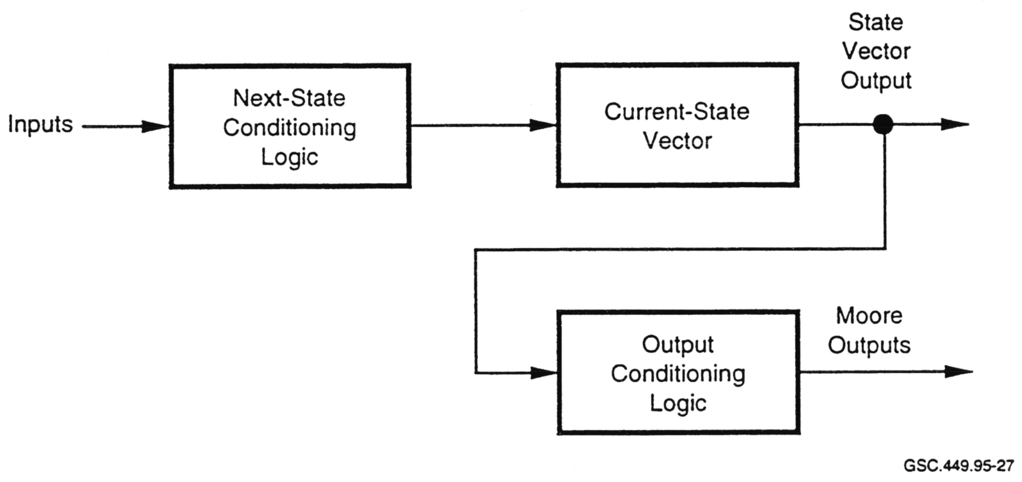


FIGURE 27. MOORE STATE MACHINE

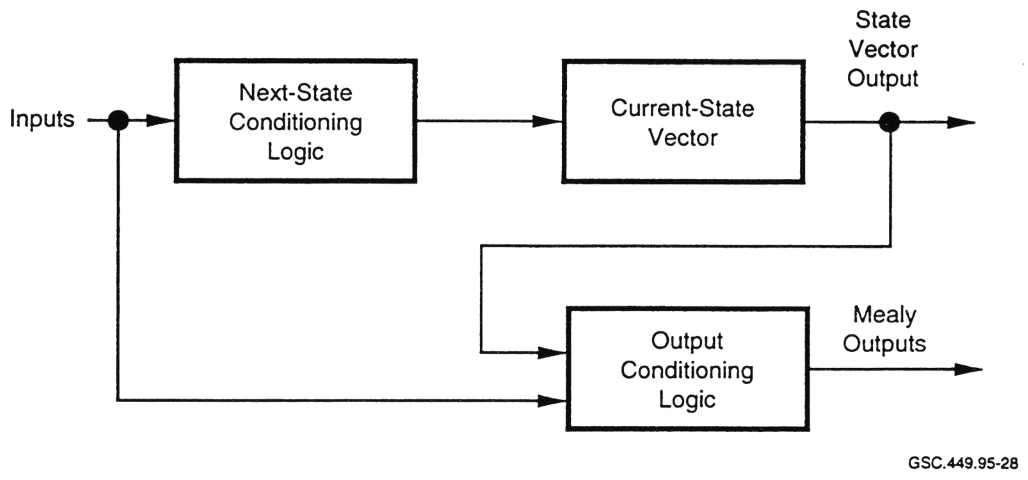


FIGURE 28. MEALY STATE MACHINE
(Rajan 1995)

Both graphical and text methods are used by current tools to capture designs. Although state machines can be performed manually for small designs, the automated design tools available can reduce the number of errors introduced by the "pencil and paper" approach. Tools with specific constructs for designing state machines exist and facilitate the design process, even for small designs.

State machines can be divided into three functional blocks. They are the next-state conditioning logic, the current-state vector, and the output conditioning logic. A Moore state machine has outputs that are a function only of the current state. A Mealy state machine has outputs that are a function of the current state and the inputs. Some designers combine the attributes of the two state machines to form a Mealy-Moore state machine. Usually, designers require state machines that have one or more outputs that are a function of the current state and the input variables, resulting in a Mealy state machine design.

As a design technique, the state machine is one of the most widely used. Warmke (1995) describes an ASIC with 70,000 gates, three embedded dual-port RAMs, a single-port RAM, and a number of state machines. The largest of the state machines in the ASIC had 256 states and took 4,000 lines of HDL code to implement.

The ARINC 629 protocol, used in flight-critical systems of the Boeing 777, is implemented in an ASIC using the state machine design technique. Each line replaceable unit (LRU) on the ARINC 629 bus communicates through this ASIC. There is no other means of bus access.

4.3.3 Programmable Logic Device Languages.

When PLDs were developed, manufacturers needed to develop a means for designers to express the logic that they required. Initially, designers used data sheets of a particular PLD to specify the connectivity of the interconnect matrix. Tools became available that allowed PLD designs to be expressed as logic equations. Figure 29 shows how a 4 to 16 active-low decoder may be expressed.

In this figure, Q0-Q15 are output lines and A-D are input lines.

```
/Q0  = /D * /C * /B * /A
/Q1  = /D * /C * /B *  A
/Q2  = /D * /C *  B * /A
/Q3  = /D * /C *  B *  A
/Q4  = /D *  C * /B * /A
/Q5  = /D *  C * /B *  A
/Q6  = /D *  C *  B * /A
/Q7  = /D *  C *  B *  A
/Q8  =  D * /C * /B * /A
/Q9  =  D * /C * /B *  A
/Q10 =  D * /C *  B * /A
/Q11 =  D * /C *  B *  A
/Q12 =  D *  C * /B * /A
/Q13 =  D *  C * /B *  A
/Q14 =  D *  C *  B * /A
/Q15 =  D *  C *  B *  A
```

FIGURE 29. IMPLEMENTING A 4 TO 16 ACTIVE-LOW DECODER

This is a simple example using only AND, OR, and NOT functions. Expressions can specify sequential logic and be more complex in form. PALASM, CUPL, and ABEL are common logic design tools that support PLD design and programming.

Designing in the PLD environment with such tools keeps the designer much closer to the actual hardware implementation. Available tools both automate and document the design. However, the designer is limited with this programming method to smaller designs. Much greater expressiveness is required in order to cope with designs of greater complexity. This is the case when designs require the logic density provided by devices such as FPGAs.

4.3.4 Very High Speed Integrated Circuit Hardware Description Language.

The Department of Defense mandated the use of VHDL in 1987. It became a government standard when it was made part of the Federal Information Processing Standard (FIPS) Publication 172. Computer and communication systems are designed to these standards, and as of January 1993, all digital systems supplied to the government are required to be produced in VHDL. Adoption of this standard is intended to reduce production times and life cycle costs for digital systems procured by the government.

The F-22 Advanced Tactical Fighter is the first platform designed under a United States Air Force mandate for all systems to use VHDL for top-down design. In this, and other complex systems, it is necessary for many contractors to share design data. Errors encountered by one contractor interpreting a specification differently from another can severely impact project cost and schedule. Modeling complex systems, such as exist on the F-22, in an HDL that can express design intent clearly, makes design changes easier to distribute, and can make complex designs easier to manage.

High-level languages allow the designer to express operations in several different ways. VHDL language constructs exist in three levels of abstraction: structural, data flow, and behavioral. Examples of these constructs are given in table 5.

TABLE 5. VERY HIGH SPEED INTEGRATED CIRCUIT HARDWARE DESCRIPTION LANGUAGE CONSTRUCT EXAMPLES

VHDL Construct	VHDL Code

Structural	begin
	  U1 : half_adder port map (X, Y, a, b);
	  U2 : half_adder port map (c, Cin, c, Sum);
	  U3 : or_gate port map (a, c, Cout);
	end;

Data flow	begin
	  S <= X xor Y after 10 ns;
	  Sum <= S xor Cin after 10 ns;
	  Cout <= (X and Y) or (S and Cin) after 20 ns;
	end;

Behavioral	begin
	  wait on X, Y, Cin;
	  N:=0;
	  if X ='1' then N:= N+1; end if;
	  if Y ='1' then N:= N+1; end if;
	  if Cin ='1' then N:= N+1; end if;
	  Sum <= sum_vector(N) after 20 ns;
	  Cout <+ carry_vector(N) after 30 ns;
	end;

4.3.4.1 Advantages and Disadvantages of VHDL.

Schematic entry methods are more cumbersome as gate counts increase. HDLs have made the designer's task easier for the following reasons:

- Changes are more easily made on a computer than on paper.
- HDLs isolate the designer from constantly changing technology.
- Designs can be expressed architecturally and behaviorally allowing the HDL synthesis tool to complete the design.
- HDLs allow for hardware reuse and extensive use of libraries.
- HDLs allow for standardization among different vendors.

There are also drawbacks for VHDL.

- While VHDL does allow for standardization, it currently is hindered by a lack of uniformity among various vendors, such as in features that may or may not be implemented in the vendor's particular version of VHDL.
- The different subsets of VHDL used by various vendors hinder the language's portability and ease of design migration.
- VHDL handles gate-level complexity. It does not handle gate-level timing. It is easy to make mistakes with signals arriving late from other VHDL entities.

Warmke (1995) states that designers should "have a good idea of what the lines of code really mean in terms of hardware." Every item has some associated cost, whether it is related to power consumption, area, or timing. While engineers design using top-down techniques, the building of the IC is also a bottom-up process. Timing issues need to be reflected in the top levels of the design hierarchy. Top-down design needs to account for and often be specifically tailored to satisfy lower-level timing requirements. Whatever HDL and tool suite is used, it should assist the designer at the top level to express designs that meet the requirements at the lowest level.

4.3.4.2 VHDL-Based State Machine Example.

While state machines have traditionally been designed using state diagrams, followed by the translation of the diagram into a schematic, other methods are possible. One such method is to translate a state diagram into an HDL. State machines are designed using PALASM, CUPL, ABEL, Verilog HDL, VHDL, and others. Some HDLs support the design of state machines with special commands and syntax. For VHDL, a style and syntax that is portable is necessary to cover a number of VHDL platforms. Following is an example and some design considerations for a VHDL-based state machine design.

There are several steps necessary to implementing a VHDL-based state machine. The first step is the same, regardless of the technique used. Some method of representing the states, conditions for transitions, and the next state is required. This has traditionally been done on paper by drawing bubbles and connecting arcs or by using some type of flow chart notation. If the state machine becomes large, this method can be cumbersome. The necessary information can also be entered directly into a computer using vendor-supplied tools that are designed for state machine implementations.

Defining the input and output signals is the next step. The state machine is treated as a black box and is characterized as an entity-architecture pair. The entity describes the interface of the black box to the other systems while the architecture consists of the black box contents which define the behavior of the entity. An entity description, therefore, would consist of a listing of all input/output pins.

The third step involves the definition of the states of the state machine. Enumerated data types are used (for VHDL, these are types that are user defined) to define the states. The final step is to write down the body of the state machine.

Since the state machine is characterized by three basic parts, creating the state machine body involves writing three processes which execute in parallel. These processes are

- a process for combinatorial logic that executes the next-state logic,
- a process for sequential logic that creates current-state variables, and
- a process that will form the output logic.

In order to define the process used for determining the next-state, knowledge of the current-state and all inputs is necessary. Rajan (1995) recommends the use of the case statement to implement the next-state logic. The implementation is shown in figure 30. This code is based on a state machine having five states and 11 next-state logic conditions.

This figure shows the basic structure used to define the process for determining the next-state. The five states that are coded and modified by the process are IDLE, B_Busy, S_Data, Turn_Ar, and Back off. For brevity, figure 30 shows only the code for transitions out of the IDLE and Turn_Ar states. Code is executed sequentially, seeking a match for curr_st using the *when* statements. When the match is found, the status of the input variables is then used to determine what the next state will be.

```
NxtSt: process(curr_st,FRAME_L,RDY_L,TRDYi_L,IRDY_L,STOP_L,hit,ready,term,L_lock_l,LockFSMFree,LockFSMLocked)
begin
        next_st <= IDLE;
        case curr_st is
                when IDLE | TURN_AR =>
                        if FRAME_L = '0' then
                                next_st <= IDLE;
                        else
                                if (hit = '0') then
                                        next_st <= B_BUSY;
                                else
                                        if (term = '0' or (term = '1' and ready = '1')) and
                                          (LockFSMFree = '1' or LockFSMLocked = '1' and
                                            L_lock_l = '0')) then
                                                next_st <= S_DATA;
                                        end if;
                                        if ((term = '1' and ready = '0') or (LockFSMLocked = '1' and
                                          L_lock_l = '1')) then
                                                next_st <= BACKOFF;
                                end if;
                        end if;
                end if;
        end case;
end process;
```

FIGURE 30. NEXT-STATE DETERMINATION CODE (RAJAN 1995)

Creating the current-state variables is a sequential logic process. This is done in two different ways. First, upon reset, a state machine is set to a known, or idle, state. The other is to generate the current-state condition based on the state machine clock.

Forming the output logic is the final process for the state machine. This process is performed by executing a number IF-THEN-ELSE statements. The current state and input status information is used by the process which then assigns values to the output pins.

There are a number of points to remember and guidelines to follow, even in a limited application of VHDL such as this. When the case statement is used, it is necessary to specify all possible cases. This means that a corresponding "when" condition must exist for all possible states. It is possible that a compiler may not flag case statements that are incomplete.

Also, it is recommended that a sequence of statements similar to the following appear at the end of the case statement:

```
when others =>
next_st <= IDLE.
```

This ensures that if unspecified states do exist, they will be handled by the WHEN OTHERS statement. Also, changes can be made to the case statement at a later time in a product's life cycle.

When states are eliminated, the unspecified states are then handled by the WHEN OTHERS statement.

The if-then construct should be terminated with the ELSE statement. In VHDL, memory elements are associated with each signal. If the signal is not assigned, the compiler may store its last value. Some compilers will generate an associated memory element in order to implement storage of this value. It is better to close each IF-THEN with the ELSE statement since it will not only make the code more readable and portable but will also prevent the compiler from assigning extra memory elements.

Also a consideration when using HDLs to create state machines is the manner in which the hardware will be encoded. The encoding chosen by the HDL may not be the preferred or optimal choice for every application. There are three common types of state machine encoding: traditional, multiplexer, and one-hot. Using the one-hot method generates higher performance and simplifies state decoding. Where portability or design style is a concern, designers also may choose to encode states using the predefined data type **bit_vector**. This encoding, in conjunction with a sequence of IF-THEN-ELSE statements can implement the one-hot architecture.

State machines can be further optimized using a "don't-care" specification where possible. This allows the synthesis tool to simplify the circuit. One way to do this is to use the data type **std_ulogic_vector**. However, all compilers may not offer this type (Rajan 1995).

4.3.5 Verilog Hardware Description Language.

Verilog HDL is the most widely used HDL. Verilog HDL started as an input language for the Verilog logic simulator. In 1991 the language was put into the public domain and many vendors now provide synthesis and simulation tools based on Verilog HDL.

Verilog HDL is a structured language that has capabilities similar to VHDL. Vendors support Verilog HDL with libraries and tools that accept it as input for simulation and synthesis. Although it is not standardized as is VHDL, there is an Institute of Electrical and Electronic Engineers (IEEE) committee that has been working on adopting Verilog HDL as a standard.

An example of how a simple circuit is coded in Verilog HDL is given in figure 31.

Once a module is defined, it may be used repeatedly. In figure 31, there are six input signals defined and one output signal. Signals f, g, h, and i are intermediate logic values. The ASSIGN statement specifies the equations required to produce the intermediate logic values. The operators are as follows:

& is the AND function
~ is the NOT function
| is the OR function

```
module example (out, a, b, c, d, e, clk);
input a, b, c, d, e, clk;
output out;
wire f, g, h, i;
reg out;
parameter delay = 1;

assign f = a & b & d;
assign g = ~a & c | f;
assign h = c | d & e;
assign i = (g & ~h) | (~g & h);
always @ (posedge clk);
out = #delay i;
endmodule
```

FIGURE 31. VERILOG HARDWARE DESCRIPTION LANGUAGE CODING EXAMPLE

When this module executes, the positive edge of the clock causes the value of “i” to appear on the output “out” after a programmable delay period which represents the signal propagation time through the D-type flip-flop.

Many designers hold that Verilog HDL is easier to learn and use than VHDL. Arguments over which HDL is better have been ongoing for years. Essentially, the advantages and disadvantages for VHDL outlined in section 4.3.4.1 also apply to Verilog HDL.

4.3.6 Analog Hardware Description Languages.

While HDLs for digital systems are commonplace and some have been standardized for a number of years, at the time of this publication, there is no official analog hardware description language (AHDL) standard. There is an IEEE standards committee working on an analog extension standard, VHDL 1076.1. This extension is commonly known as VHDL-A. Also, some interest exists in proposing an analog extension to Verilog (Verilog-A) for possible adoption as an IEEE standard.

System-level analog design and simulation would benefit from the availability of an analog HDL standard in the same way that HDLs for digital systems benefit designers. The availability of a library of behavioral analog models that are fast, accurate, and robust is required. Analog synthesis, which is a driving force for the development of AHDL, is still in the early stages of development.

Simulation was one of the early needs for the analog designer. Computer Analysis of Nonlinear Circuits, Excluding Radiation (CANCER) was one of the original nodal analysis programs developed in 1969-1970. This program evolved into Simulation Program with Integrated Circuit Emphasis (SPICE), then into SPICE2. SPICE, along with its derivative programs, is likely the most widely used circuit simulator.

There are problems facing AHDL development that make it more difficult than HDL development. The analog language must be able to describe noise, complex statistics, and second and third order effects, as well as other unique analog characteristics. Accounting for these complex effects is necessary to make the AHDL design useful. Practical top-down analog design requires an accuracy of five to 20 percent of the SPICE solution (Coston 1994).

4.4 SYNTHESIS OF DIGITAL LOGIC.

A great amount of research has taken place and is still ongoing in the area of logic synthesis. While a number of synthesis tools are now available, further research is needed in specific areas. Synthesis tools are a necessity for dealing with the complexity of modern digital design. Without automation tool support, ASIC design today would be quite impractical. Synthesis tools have increased design efficiency significantly. Designers are able to spend more time describing the design and setting various parameters and constraints using synthesis tool scripts allowing the tool to handle the more tedious portion of the design.

While design efficiency is increasing, device line widths are decreasing, and geometries and metal layers are increasing. This leads to yet higher levels of device complexity with no end in sight. Tools often are in the "catch-up" mode allowing designers simply to cope with this complex technology but not to manage it efficiently as in the case of a mature tool suite. High-level synthesis techniques, addressed in section 4.4.4 seek to address this problem.

When considering the number of gates that can be designed into an ASIC, the schematic design entry method has little practical value for most current designs. This method is a disadvantage for larger designs. Only where the number of gates is small can a schematic-based design be considered practical. While schematics or netlists are still a part of the design process, they are created by the synthesis tool after the gate-level design is created. In the move away from hand-generated, schematic-based designs, disadvantages include:

- Schematics that are automatically generated can be difficult to read.
- Schematic changes are not reflected back into the HDL.
- Small changes to designs that are schematic-based are easier to make.

A schematic capture-and-simulate design methodology had been used extensively until recently. Often the design requirements are received by the designers with no guidance concerning the implementation. A block diagram may then be drawn, which serves as a preliminary specification. This block diagram may be refined further before it is passed on to a team of logic designers.

The designers convert each of the blocks into a logic schematic. This schematic is then captured using a schematic-capture tool. A simulation is run to verify timing, functionality, and fault coverage. Fault coverage refers to the capability to control and observe internal circuits of an IC. The captured schematic is then used as input for placement and routing tools.

4.4.1 Synthesis Overview.

Synthesis involves the incorporation of three processes into a single tool. These processes are HDL translation, mapping, and optimization. The translation process takes the HDL and converts it into boolean equations. Mapping is a process of synthesis whereby the defining design equations are translated into a specific technology-dependent component library or gate design. Optimization is performed in order to create a product that favors a particular characteristic, such as low power consumption, testability, area reduction, or high-speed operation.

Synthesis tools should be sufficiently flexible to allow the designer to specify boundary conditions for an optimal design. Some of the parameters and constraints that designers need to specify include:

- signal drive capability;
- rise times and fall times for various signal paths;
- maximum fan-out;
- operating conditions, such as voltage, temperature, and packaging;
- models for wire loading;
- clock data such as frequency and setup and hold times; and
- design hierarchy preferences.

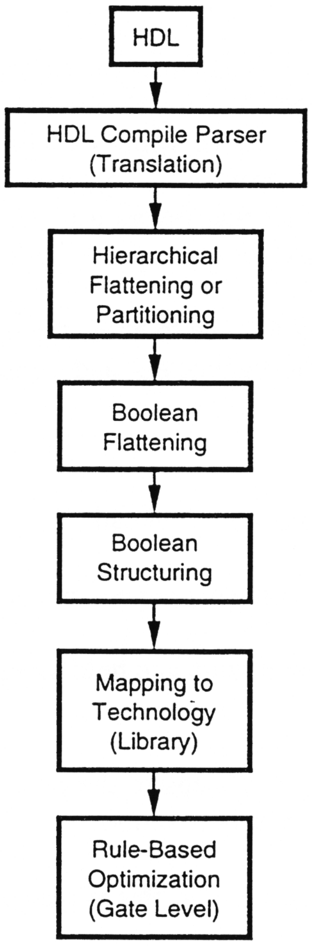
The synthesis tool performs a number of steps in the process of arriving at a gate-level design. Figure 32 shows these steps and the order of execution.

At the top level is the HDL. For large designs, this is typically implemented with Verilog HDL or VHDL.

An HDL compiler and parser is the next step in synthesis. At this step the HDL syntax is checked. A translation of the HDL description is made into high-level equation form. A high-level structure is created for use in the following steps.

Partitioning and hierarchical flattening is performed in the next synthesis step. Partitioning breaks up the design into parts that should be synthesized separately. This is done to make the design more manageable and more easily synthesized. A good synthesis tool can greatly reduce the number of interconnects necessary between different parts of the design.

Hierarchical flattening consists of merging together levels of design hierarchy in order to simplify a portion of the design and obtain a single level. Boolean flattening removes intermediate variables, minimizing the number of logic levels. Gate count is reduced and processing speeds are increased with boolean flattening due to fewer stages between inputs and outputs. Examples of hierarchical and boolean flattening are shown in figure 33a and b.


HDL
HDL Compile Parser (Translation)
Hierarchical Flattening or Partitioning
Boolean Flattening
Boolean Structuring
Mapping to Technology (Library)
Rule-Based Optimization (Gate Level)

GSC.449.95-29

FIGURE 32. LOGIC SYNTHESIS PROCESS
(Widman 1994)

Mapping is a portion of the synthesis process that relates the boolean representation to a particular component library. Component libraries are technology dependent and supplied by the ASIC vendor. As with software libraries, ASIC libraries are developed from HDL descriptions and then compiled into a binary library file. Actual library designs differ based on whether a design is based on CMOS, emitter-coupled logic (ECL), or other implementation technologies.

Libraries contain design details such as:

- functional description,
- setup and hold requirements,
- wire loading data,
- fan-out limitations,
- signal rise and fall times, and
- area.

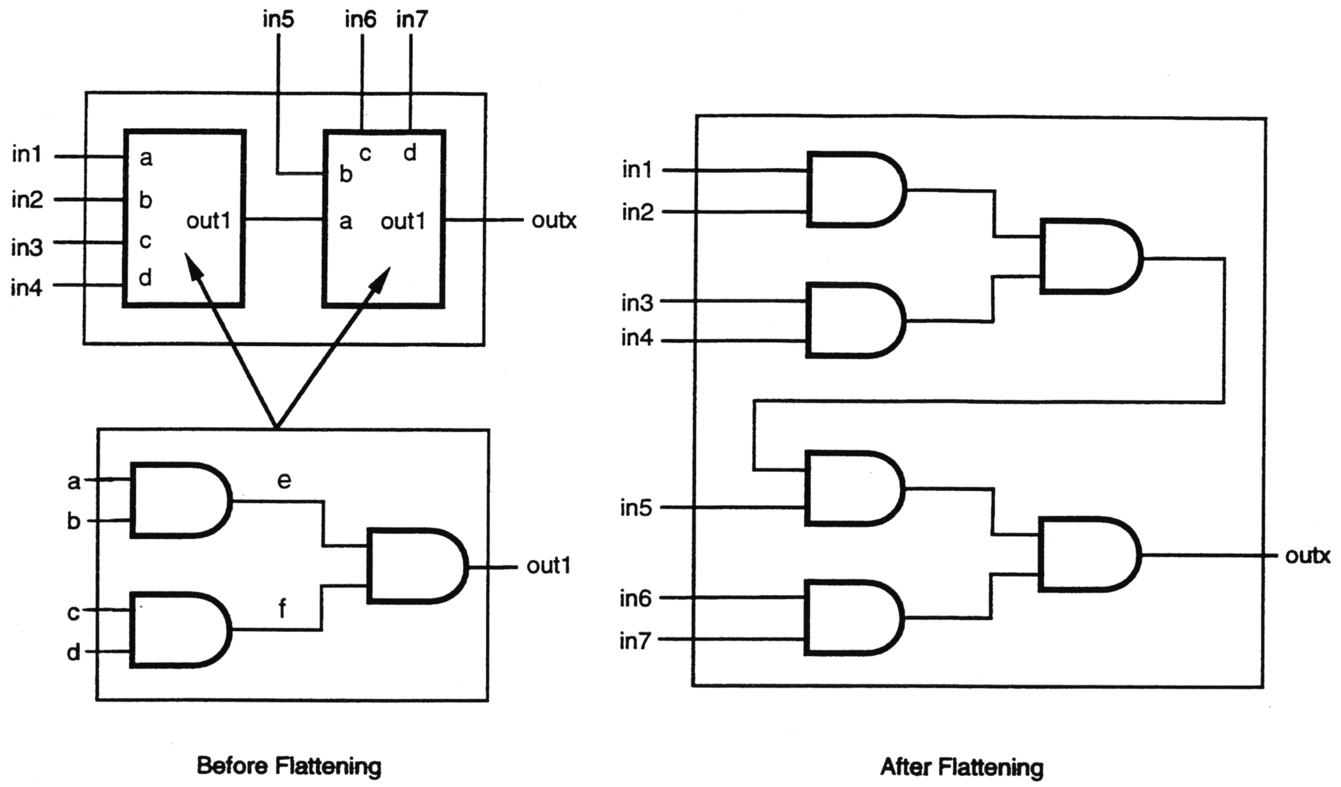


FIGURE 33a. HIERARCHICAL FLATTENING

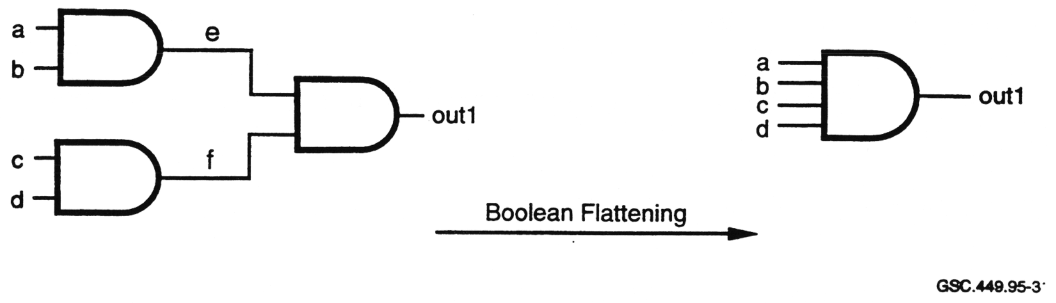


FIGURE 33b. BOOLEAN FLATTENING

Optimization occurs as the final step. This is a gate level optimization that is designed to either reduce the area used by a design or to increase the speed of a design.

Prior to synthesis, designs were performed manually. The engineer worked from a specification to create a gate level implementation. The end result is a schematic drawing which depicts the elements of logic comprising the design. When design is performed this way, optimization is both difficult and time consuming. Also, this method is tied closely to a specific technology. Changing to another technology can be time consuming.

Switching from a schematic-based to a synthesis-based design involves several methodology changes. Some old tools are no longer needed while other newer tools must be learned. A gate change is easier to do using a schematic. An equivalent change made by an HDL and then synthesized may produce other changes since the correspondence between the HDL and netlist produced as a result of synthesis can be obscure.

There is no direct correspondence between the number of statements of HDL code and the number of gates that are generated by the synthesis tool. A high-level description can generate few or many gates. Small HDL program loops can cause a large number of gates to be created. Detailed and large descriptions are sometimes reduced to a small number of gates. However, if the HDL code is broken down into groups whose code-to-gate correspondence is known, then reasonable estimates of gate counts can be achieved.

4.4.2 High-Level Synthesis.

In the last few years the level of on-chip integration has increased to such an extent that it has become impractical, if not impossible, to use the capture-and-simulate design methodology on large designs. Vendors have designed a new generation of tools based on logic synthesis. Synthesis allows the description of a circuit at a behavioral level without having to worry about any implementation details.

High-level synthesis transforms a design into a register transfer level (RTL) description. An RTL description is comprised of building blocks such as memories, registers, ALUs, data paths, and multiplexers. A general representation of the high-level synthesis tool functionality is shown in figure 34.

The various functions of the synthesis tool include:

- RTL library. This contains the component models, both physical and simulation, used by the synthesis tool.

- Netlist generator. This generates, in RTL format, the final design structure.

- Netlist. This includes RTL components and their simulation models.

- Behavioral description. This is the HDL code.

- Compiler. The compiler takes the HDL code as input and creates an internal representation of the design.

- Design Constraints. These are parameters specified by the designer that the synthesis tool requires in order to constrain the design.

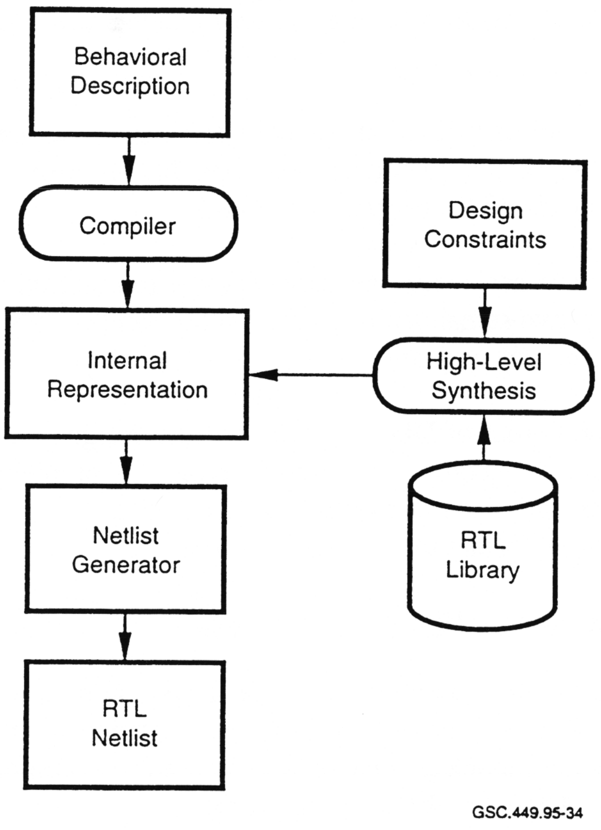


FIGURE 34. HIGH-LEVEL SYNTHESIS TOOL FUNCTIONALITY
(Gajski and Ramachandran 1994)

High-level synthesis tools allow increases in productivity since designs are performed at a higher level of abstraction. Many of the tasks formerly done manually are now relegated entirely to computers. ASICs can be modeled, synthesized, simulated, and debugged using synthesis-based tools on workstations and PCs.

4.4.3 Input Methods for Synthesis.

Regardless of the type of input used, design intent is expressed by the input description. Typically the input description is written in an HDL and does not contain information on types of components, component interconnections, or circuit structure. HDLs may support the inclusion of design structure details. When more structural details are known at the beginning, the synthesis task is made easier. Designers are less burdened, however, if they can restrict their input descriptions to behavioral information and leave the details of structure to the tools.

Numerous HDLs have been developed in an effort to automate logic design. HDLs provide designers considerable flexibility in programming circuit functionality. Condition constructs such as CASE and IF statements are provided, as well as variables and ways to manipulate them. Loop statements provide for iterations of a set of statements. The two most popular HDLs, Verilog HDL and VHDL, provide designers with design flexibility along with the required tool support from most vendors.

Designers should take note that different styles of HDL code can be implemented with different synthesized results. It is possible to code expressions that are semantically equivalent in different styles. According to Gajski and Ramachandran (1994):

> Most high-level synthesis tools are very sensitive to description style. Two designs synthesized from two semantically equivalent but syntactically different descriptions may differ significantly in quality.

Variations in the type and order of HDL constructs affect design quality. This problem is referred to as syntactic variance. Two methods are used to avoid this problem. One is to restrict the input description style so that variations cannot occur. The other is to use a tool to convert all input descriptions into a common format for the synthesis tool.

4.4.4 Internal Representation for High-Level Synthesis.

As mentioned previously, the compiler takes the HDL code as input and creates an internal representation of the design. The type of internal representation should be chosen so that it closely matches the problem being modeled. One technique used to accomplish this representation is the data flow graph (DFG). A DFG is shown in figure 35.

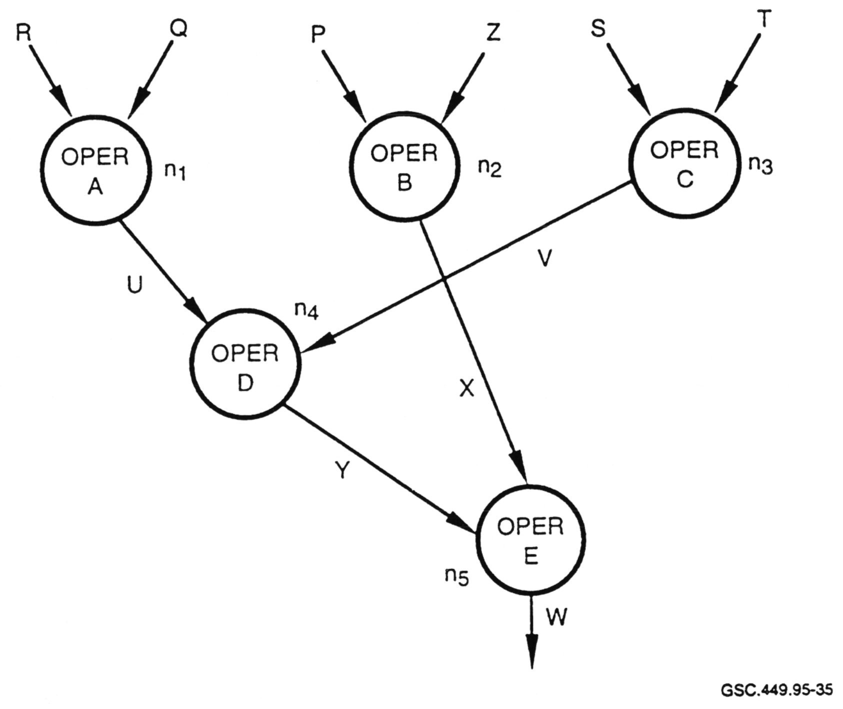


FIGURE 35. DATA FLOW GRAPH

The DFG consists of a set of nodes and arcs that interconnect them. The nodes represent an operation that is defined in the input description. An operation can be a shift left, add, subtract, or any of a number of operations that are defined in the HDL. Each operation of the input description has a corresponding node in the DFG. Nodes are connected by arcs. An arc U exists between n_1 and n_4 since Operation D is dependent upon receiving the results from Operation A. Likewise, the arc V exists between n_1 and n_3 since Operation D is also dependent upon receiving the results from Operation C before it can execute.

The DFG works well for representing data dependencies but cannot handle actions that control the data flow based on external conditions. For this, a control-data flow graph (CDFG) is used. Figure 36 shows how a CDFG is used to control the flow of data.

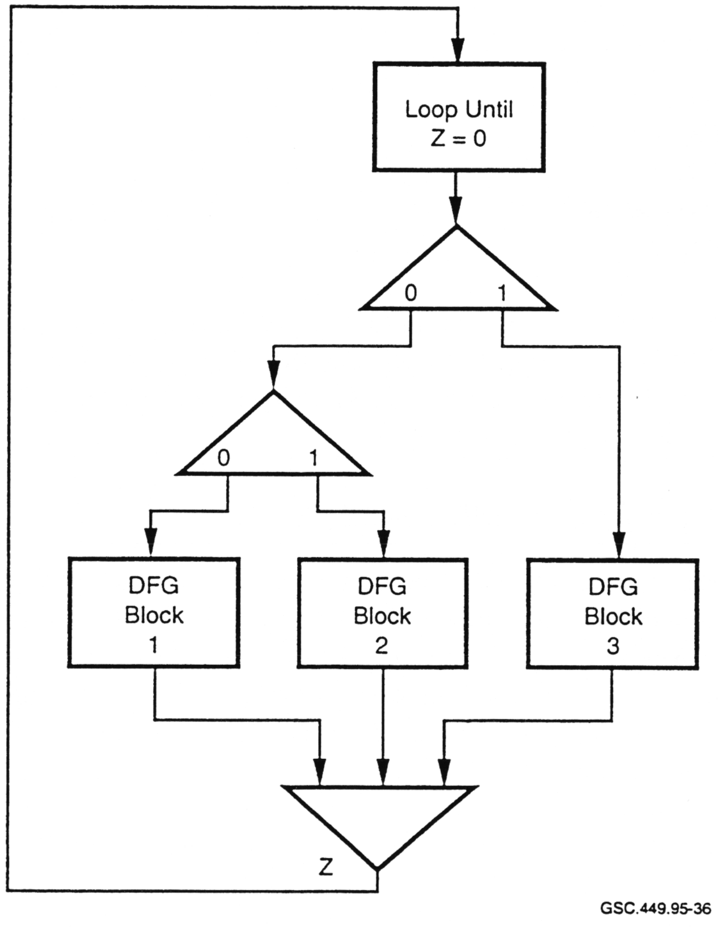


FIGURE 36. CONTROL-DATA FLOW GRAPH

External conditions as well as data dependencies are represented by the CDFG. External signals, timers, and looping are examples of external controls for the CDFG. Special nodes are used in the CDFG to represent these external controls. The CDFG consists of a combination of data dependency blocks and the control structure.

Each block specified in the HDL is represented by data dependency blocks in the CDFG. Also, the control structure specified by the designer in the CDFG representation is maintained. Data dependencies are represented only within the blocks of the CDFG. Where the synthesis tool works from the CDFG, the order of execution of the blocks is maintained. A disadvantage for these types of synthesis tools is that even if there are no data dependencies in two sequential blocks of the CDFG, the order of execution is still maintained. An improvement in the synthesis tool efficiency can be realized by allowing the order of execution to be based on data dependencies and by eliminating user-defined control constructs.

4.4.4.1 The Finite State Machine Model.

Synthesis of sequential circuits is based on the finite state machine (FSM) model. For high-level synthesis, variables and data paths are incorporated into the FSM model. There are five basic parts to the FSM.

- input values
- output values
- a set of states
- next-state computation
- output computation

The FSM has been used extensively in numerous applications. As the number of states increases, the designer's ability to manage and comprehend the design diminishes. While simple designs can consist of less than 10 states, even low complexity designs such as I/O interfaces can contain several thousand states.

4.4.4.2 The Finite State Machine with Data Path Model.

In order to manage more complex designs, the FSM with data path (FSMD) model has been developed. This model has been implemented on synthesis tools to allow designers to cope with and automate increasingly complex designs. The FSMD uses storage elements such as memories and registers for storing variables that represent different states of the FSMD. An n-bit register can represent up to 2^n different states. An eight-bit register can represent up to 256 different states, and a 16-bit register up to 65,536 different states.

The FSMD is a model that can represent all digital designs. Designs that are largely control-oriented or data-oriented can be accommodated equally well. A control-oriented design handles large control functions with perhaps a small data path. A data-oriented design is focused largely on data path operations, with a smaller portion dealing with control issues.

The FSMD consists of a set of variables, storage expressions, status signals, and storage assignments. Similar to the FSM, the next state and outputs of the FSMD are computed based on the current state and external signals. However, the FSMD also uses internal status signals such as relationships between two data path values in this computation.

A technique called pipelining is used in the FSMD to increase design performance. There are three types of pipelining: component, data path, and control. Component pipelining increases the utilization of functions that are within the data path. This is done, for example, where an ALU is required. An ALU in the data path allows for resource sharing. In certain applications, the same sequence of operations is performed on an input data stream. Data path pipelining is used for operations that execute repeatedly in the data path. Control pipelining is used since the control operations are repeated in each state of the FSMD. Control operations include computing control signals, variable values, and the next state.

4.4.4.3 Data Path Example.

A specification is refined into a structural design consisting of standard components. A design style and specific architecture are selected. The design style reflects the main qualities, or features of the design. These features include items such as direct memory access (DMA), serial I/O, and instruction pipelining. The architecture further defines the design in terms of unit building blocks, their characteristics, and their interconnections. In defining an architecture for a processor, for instance, building blocks would include types of conditional branches, the number of general and special purpose registers, the number of pipeline registers, data paths, status flags, and so on.

Design quality is influenced by algorithms used by synthesis tools. Synthesis tools focus upon specific target architectures. The target architecture should "match" the design. Sophisticated algorithms are required by synthesis tools, especially in light of the current IC densities.

Gajski et al. (1992) give an example of how design quality can be influenced by the synthesis tool. Figure 37 shows a portion of a design using three buses, some registers, and an ALU.

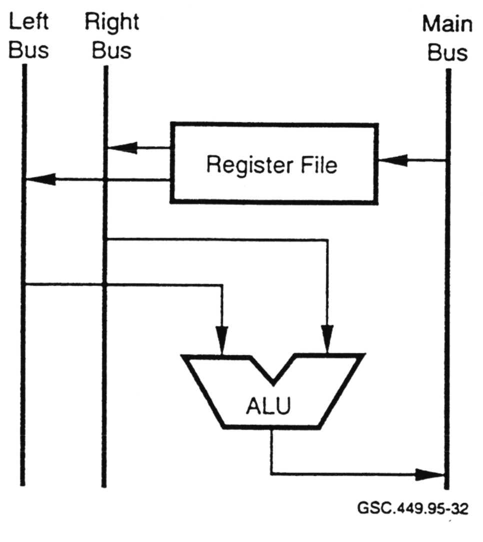


FIGURE 37. A THREE-BUS DESIGN IMPLEMENTATION

For this architecture, two operands can be accessed concurrently from the registers, operated upon, and the results placed on the main bus and stored back in registers within one clock cycle.

If A, B, C, D, E, and F are available registers, then the operation:

$$\begin{array}{ll} E \Leftarrow A - B & \text{(Clock t1)} \\ F \Leftarrow C + D & \text{(Clock t2)} \end{array}$$

can be performed in two clock cycles.

Since data paths consume silicon, an approach using only two data paths is preferred. Also, the ALU and registers are used only during a portion of the clock cycle. When one is active, the other is inactive. Hence, improvements can be made by altering the design slightly.

Figure 38 shows one technique used to enhance performance. By adding the pipelined left instruction register (LIR) and right instruction register (RIR), cutting the clock period by one-half, and changing the scheduling algorithm, the sequence of execution then becomes:

$$\begin{array}{ll} LPR \Leftarrow A;\ RPR \Leftarrow B & \text{(Clock t1/2)} \\ E \Leftarrow LPR + RPR;\ LPR \Leftarrow C;\ RPR \Leftarrow D & \text{(Clock t2/2)} \\ F \Leftarrow LPR - RPR & \text{(Clock t3/2)} \end{array}$$

While the overall execution time is faster, the number of data paths has been reduced by one, saving a significant amount of silicon area.

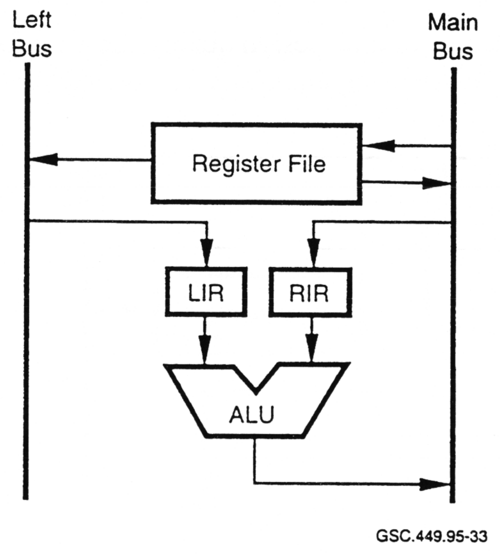


FIGURE 38. A TWO-BUS DESIGN IMPLEMENTATION USING PIPELINING

4.4.5 Tasks of the Synthesis Tool.

One of the tasks of synthesis is to shorten critical timing paths. This can take a number of design iterations. Where high-level synthesis is unable to address critical timing paths adequately, other approaches can be implemented by the designer. Two methods are used by designers. One provides the synthesizer with the optimum set of constraints and environmental data (signal arrival times, loading factors, etc.). The other involves restructuring the hardware by modifying the VHDL code. What will actually work is design-specific. Ways that this type of problem can be mitigated include (Warmke 1995):

- restructure the RTL,
- move critical logic into an earlier or later pipe stage,
- duplicate the logic in different entities of the IC to avoid long paths,
- add pipeline stages,
- reduce entity sizes by adding subhierarchy or new entities, and
- reimplement or redesign algorithms.

4.4.5.1 Allocation.

One of the decisions that designers face is whether to optimize a design for cost and area or for performance. Where an architecture contains parallelism, there is opportunity to share resources. Doing so saves on area, allowing more functionality per unit area, and reduces cost by allocating fewer resources. Operations are sequential and valuable area is spared for other functionality. On the other hand, allocating more resources where parallelism exists allows for greater performance.

In order for comparisons to be made, the tool must make an approximation of the area and performance values. The physical models stored in the tool's RTL library can provide accurate figures for comparisons. Table 6 shows what kind of data can be obtained from the RTL library.

TABLE 6. COMPONENT VALUES FROM REGISTER TRANSFER LEVEL LIBRARY (Gajski and Ramachandran 1994)

Component	Delay (ns)	Area (μm^2)
Fast ALU	20	500
Slow ALU	70	300
MAX	80	700

The ALU, which performs basic mathematical computation, can be a fast design, taking up more of the available geometry, or can be a slower design, using significantly less area. For the ALU, 70 ns and 20 ns versions are available. By design, the 70 ns ALU uses parallelism and takes up less area than the 20 ns ALU.

Using the information from the RTL library as shown in table 6 and the CDFG, estimates on area and performance can be made. If one MAX function and three ALUs are required in the CDFG, the possible constructions are shown in table 7.

TABLE 7. POSSIBLE CONSTRUCTIONS FOR SPEED AND SIZE TRADE-OFFS

Construction	No. of 70 ns ALUs	No. of 20 ns ALUs	No. of MAX (80 ns)	Area (μm^2)
A	0	3	1	2,200
B	1	2	1	2,000
C	2	1	1	1,800
D	3	0	1	1,600

From the table, it can be seen that if slower performance is acceptable, a reduction in the area can be made of up to 600 μm^2. Using the information from this table, the delay times, and the CDFG, it is possible to construct a performance trade-off curve. Each construction, A through D, will have a different area and a different delay. Designs with the best performance will also consume the largest area. What is acceptable will depend on the application. Trade-off curves can also be obtained for other portions of the design. These can include interconnection units, storage units, and storage unit ports.

While automation is essential, certain parts of a design still require tailoring to meet specific requirements. Synthesis tools should allow the designer to allocate the mix of hardware resources that will produce the best design. They should also provide the designer with accurate values for parameters such as area and performance so that initial designs do not have to be reworked with data based on some later part of the design process. In the future, tools should provide for exploration of more complex architectures (Gajski and Ramachandran 1994).

4.4.5.2 Scheduling.

When a synthesis tool performs scheduling, it assigns operations and memory accesses to clock cycles. The tool uses one of two algorithms for scheduling; it uses either a time-constrained scheduling or a resource-constrained approach. Since the user specifies the clock period for the device, the scheduling algorithm seeks to generate the best performance, or maximum number of operations per clock cycle, for a given approach.

4.4.5.2.1 Resource-Constrained Scheduling Approach.

When the user specifies all the resources, the tool seeks to maximize the usage of these resources within as few clock cycles as possible. This is known as the resource-constrained approach. In this approach, operations are typically scheduled one state at a time. Two guidelines that the tool follows are that resource constraints are not exceeded and data dependencies are not violated. Data dependency can be violated if data are required for an operation, but that data is not available since the operation in which it is produced has not yet been executed. Resource constraints can be exceeded if an operation is required in a given state and the particular resource is unavailable. For example, scheduling two ALU operations in the same state when only one ALU is available is a violation of resource constraints.

4.4.5.2.2 Time-Constrained Scheduling Approach.

If the overall performance is known (i.e., the number of control steps or clock cycles) but the resources are not completely specified, then the scheduling algorithm seeks to generate a design with the least amount of functional units. This is the time-constrained approach.

A time-constrained algorithm essentially performs the following steps:

- Computes the earliest and latest control step in which an operation can take place.
- Estimates the maximum number of functional units that must be scheduled between the earliest and latest control steps (this gives an indication of the cost).
- Evaluates the cost of scheduling an operation in each of the control steps in which it will work.
- Selects the lowest cost by minimizing the number of functional units.

Since area and delay characteristics vary for the same component, there may exist multiple implementations in the RTL library. The task of a scheduler is to ensure that critical paths are assigned the faster implementations while noncritical paths receive the slower implementations. This will ensure performance goals are met and costs are kept as low as possible. Figure 39 shows a possible schedule based on allocation A of table 7.

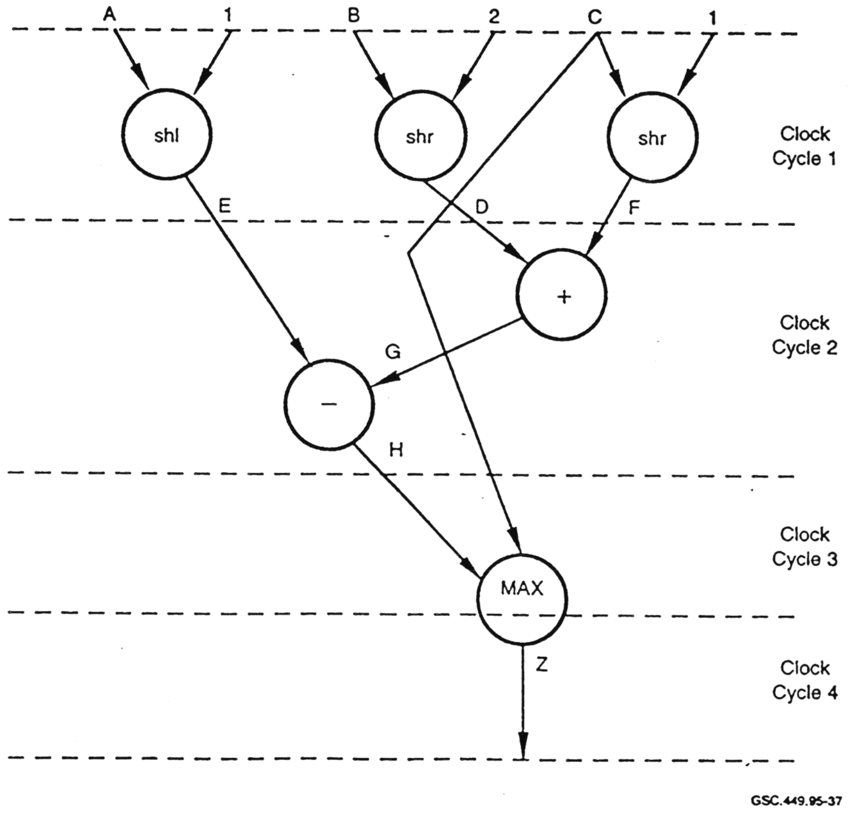


FIGURE 39. SCHEDULER EXAMPLE

Assuming a clock period of 60 ns and that this particular operation is in a critical path, all three of the ALUs are required to be the 20 ns versions, while 80 ns is the only version available for the MAX function.

4.4.5.3 Binding.

Operations and memory accesses are assigned to available hardware units in the binding phase. Resources such as functional, interconnection, or storage units can be shared by different operations, data transfers, and accesses as long as the resources are mutually exclusive. Two operations are mutually exclusive if they do not execute simultaneously. When this is the case, these operations can be bound to the same hardware unit.

Three types of binding are performed. They are

- storage binding,
- functional unit binding, and
- interconnection binding.

Storage binding assigns variables to storage units, such as registers and memory units. If two variables are not active simultaneously in the same state, they can be bound to the same memory or register. Each operation within a control step is assigned a functional unit in functional unit binding. Finally, for each data transfer among the various units, an interconnection unit is assigned during the interconnection binding process.

Storage binding, which occurs after states are assigned to all operations, partitions variables into compatible groups. A group of variables is compatible if they are not active at the same time. Compatible groups are determined by how long a variable is active over a set of control steps. Table 8 indicates the variable activity for the scheduler example of figure 39.

TABLE 8. ACTIVE VARIABLES FROM FIGURE 39

VARIABLES	D	E	F	G	H
CLOCK CYCLE 1					
CLOCK CYCLE 2	X	X	X	(X)	
CLOCK CYCLE 3					X
CLOCK CYCLE 4					X

From this table, it can be seen that variables D, E, and F all require storage in clock cycle 2, making it necessary for three storage registers. Since G is generated and used within state 2, there is no need for storage of this variable. Also, it can be seen that variable H is active over two clock cycles.

A graph can now be created with nodes that represent variables and segments that represent nodes with mutually exclusive active times. Each node is connected to all of its neighbors. Figure 40a shows a graph of this relationship based on table 8.

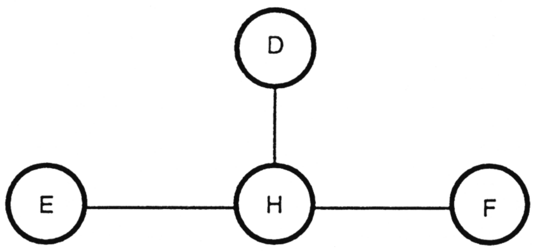


FIGURE 40a. GRAPH OF NODES AND NEIGHBORS

This graph now needs to be partitioned into cliques. A clique is a subgraph of mutually exclusive nodes and connections between them. This is shown in figure 40b. In this example, there are three possible cliques that can be chosen for implementation: E-H, D-H, or F-H. If E-H is chosen, this means that both E and H will share a common register since their requirements for variable storage occur in different clock cycles. Consequently, the nodes D and F need to have registers assigned to them. The variables D, E, and F cannot use the same registers since they are all active in the same clock cycle. The three possible solution sets are shown in table 9.

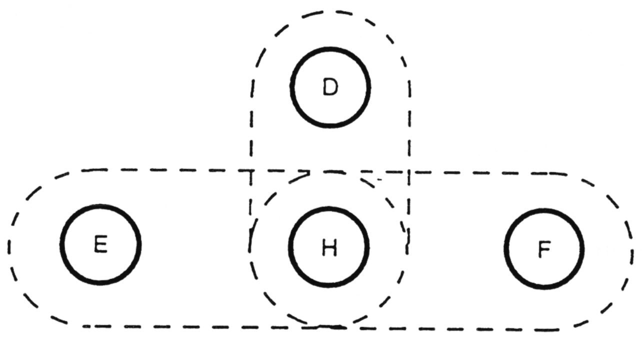


FIGURE 40b. POSSIBLE CLIQUES BASED ON 40a

After the storage binding is completed, the functional unit binding and interconnect binding are performed. Functional unit binding is performed on a cycle-by-cycle basis. Operations in the first cycle are assigned to functional units. This means that the three shift operations shown in figure 39 are each assigned to a particular ALU. This process continues for all stages until all operations are assigned. Interconnects are also assigned in conjunction with this process (Gajski and Ramachandran 1994).

TABLE 9. POSSIBLE SOLUTION SETS FOR VARIABLES

REGISTERS	STORED VARIABLES
R1 R2 R3	E, H D F
R1 R2 R3	D, H E F
R1 R2 R3	F, H D E

4.4.6 Other Design Issues for Synthesis.

4.4.6.1 Handling Synthesis Unknowns.

In hardware, there may exist uninitialized values, but unknowns do not exist. All logic settles to a "1" or "0" state. Unknowns can occur in simulation when there exist uninitialized nodes, logic contention problems, or nets where the value cannot be determined. Synthesis tools deal with values of 1, 0, and the high impedance state, but have difficulty with unknowns. Designers should know how to represent unknowns in their high-level design and how the compiler, synthesis, and simulation tools handle unknowns.

4.4.6.2 Design Strategies for Synthesis.

Consistency is one of the requirements when designing ASICs. Coding style is one of the major factors in producing designs that perform according to the designer's expectations. Some general goals for consistent techniques include (Widman 1994):

- Ensure that the design is completely documented and take steps to ensure that it is easily modifiable.
- Use a single source for HDL design upon which simulation, synthesis, and test are based.
- Optimize the HDL design for simulation and synthesis performance.
- Develop the HDL design in a way that will guide the synthesis tool to get the expected results.

Performance of synthesis tools varies based on a number of factors. If the recommendations of the tool manufacturer are not followed, optimal results will not be realized. For instance, one of the factors that affect synthesis tool performance is the amount of physical memory contained in the workstation. Development of large ASICs is memory intensive and some tools require in excess of

256 Megabytes of RAM. If the tool does not have sufficient memory, it may use memory swapping, which will significantly slow the tool's operation. Other factors that influence synthesis tool performance include:

- speed of the workstation CPU,
- type of constraints and number of constraints that are used in the optimization phase,
- whether the design is synthesized hierarchically or flat,
- whether boolean flattening is used,
- quality of HDL code,
- types of circuits described by HDL, and
- use of incremental compilation.

4.4.6.3 HDL Techniques for Synthesis.

HDL style has an effect on several important design parameters and characteristics. Some ways of describing an HDL design are better than others. Some examples of how the design style can cause design implementation differences are given in this section. Figure 41 shows how a carry chain adder can be implemented in two different ways.

In the first implementation a ripple-carry chain is formed. This implementation is slower since it requires the carry bit to ripple through the logic before a valid sum is formed. A lookahead-carry adder performs the same sum but is much faster since the carry bit is computed by the circuit and there is no ripple time required. This lookahead-carry implementation is faster but requires a larger portion of the available geometry for layout of the lookahead-carry logic.

How expressions are grouped can make a difference in how the synthesis tool implements the expression. This is demonstrated in figure 42. In the upper portion of figure 42, the tool forms a continuous chain of adders, implementing one for each "+" operator. Three levels of logic are produced by this expression. Note that by adding parentheses in the HDL code the tool changes the design structure as seen in the lower portion of figure 42. While both expressions yield the same logical results, the latter will exhibit smaller propagation delays.

Resource sharing is a method used to reduce the amount of logic the synthesis tool will generate. Figure 43 shows how resource sharing may be implemented when adder logic is required.

Resource sharing allows more on-chip logic. There is a minor timing penalty for the resource sharing implementation, however, since the adder logic follows the mux and enable_b signal.

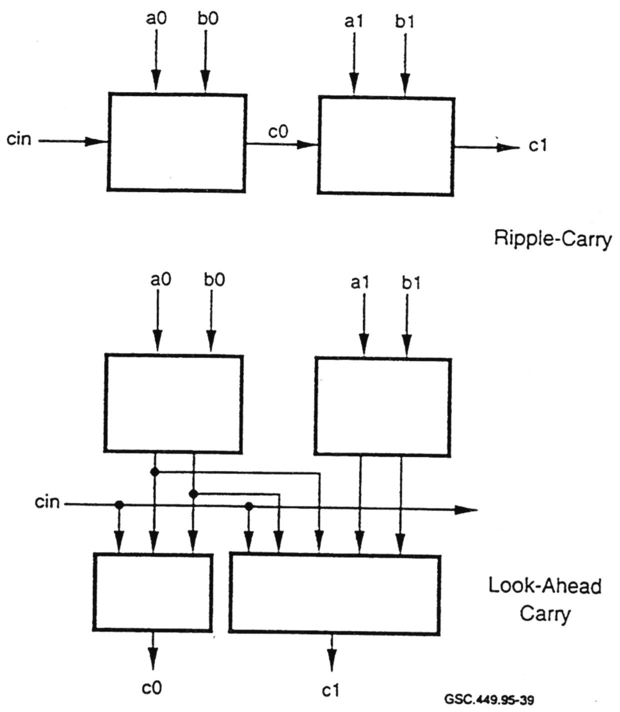


FIGURE 41. ADDER CARRY IMPLEMENTATIONS
(Widman 1994)

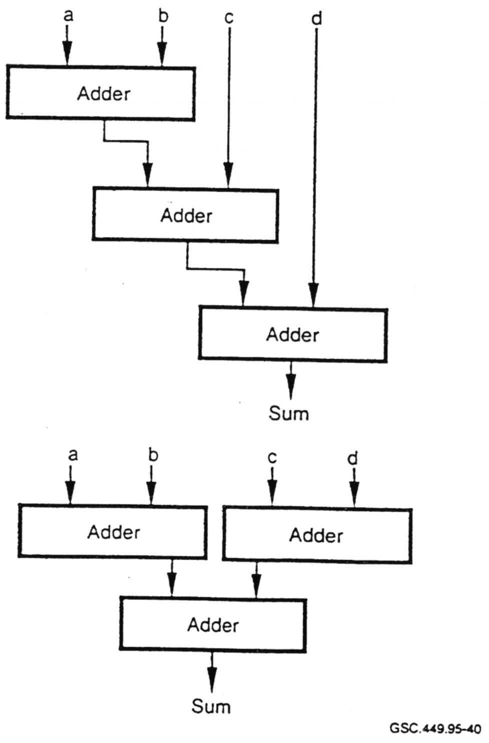
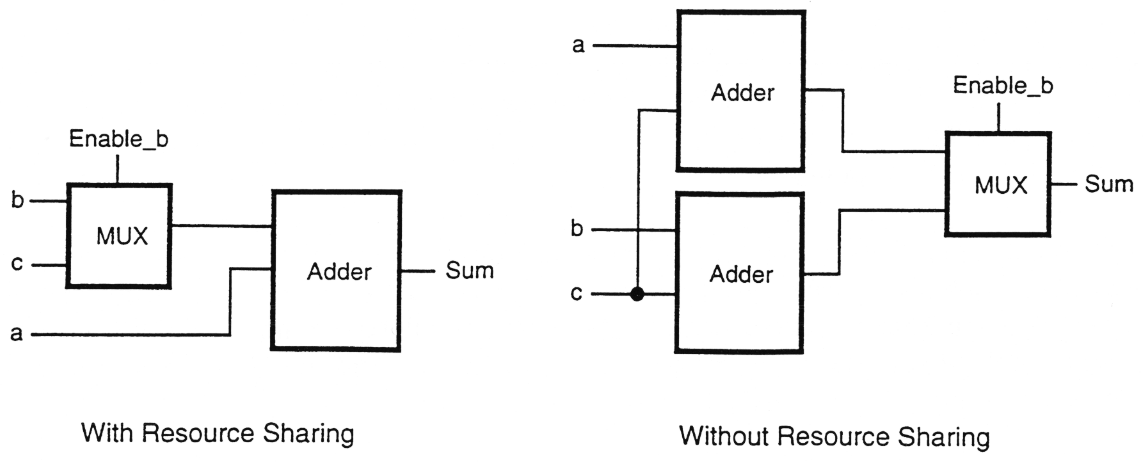


FIGURE 42. GROUPING EFFECTS ON IMPLEMENTATIONS
(Widman 1994)



GSC.449.95-41

FIGURE 43. ADDER RESOURCE SHARING
(Widman 1994)

4.4.6.4 Synthesis and Partitioning.

Designs are generally partitioned based on functionality. For a successful design, there are two things a designer must keep in mind. The first is that the size of the blocks that are synthesized should be reasonable. Both design quality and synthesizer run-times can be improved by careful selection of module size. Secondly, there need to be limitations on the number of paths that a signal can traverse. Device timing and area used can be improved by careful attention to critical timing paths. However, there is a tradeoff between timing path traversal and block size. While a synthesizer always works better on smaller modules, it also performs better when all timing paths are contained within one module (Warmke 1995). In general, partitioning a design has the following advantages:

- Designs broken into smaller parts will synthesize faster than the same design synthesized as a whole.
- Smaller designs are easier to test, debug, and modify.
- Designs with smaller synthesized parts will be easier to reuse.
- Progressive design refinement is easier to implement.
- Different constraints can be applied to smaller portions of the design, such as optimizing for speed in one part and optimizing for size in another.

4.4.6.5 Quality Measures for Synthesis.

Since synthesis is such an important part of modern ASIC design, quality measures are necessary. Two compelling arguments for quality measures exist. First, a method is needed to determine the quality of the final design. This will permit comparison of the final design with the constraints and assist in identifying critical areas in the design methodology, tool set, or design management. As an example, a high ratio of one metal layer wire count to another may indicate inferior routing or placement algorithms. Also, knowing the layout area of a design is important since it affects manufacturing yield and ultimately cost.

Second, design quality estimates are needed to influence the selection of design style and target architecture. If additional speed is needed, it may be necessary to replace a nonpipelined multiplier with a pipelined multiplier. Also, a design with two buses will require more area than a design with three buses. Quality measures give the designer a means to evaluate design alternatives, and hence, the ability to produce higher quality designs.

Two of the primary domains where quality measures support design decisions are performance and area. Performance relates to propagation time through a device and execution time. Area relates to the physical size of the design.

4.4.6.5.1 Area Measures.

Normally, area measures are divided into two categories: interconnection area and active unit area. Active units consist of functional units such as ALUs, adders, and multipliers, as well as storage units such as RAM, ROM, and registers. Multiplexers, wires, and buses comprise interconnection units.

Various methods have been used to measure the active-unit areas. Some of these methods are as follows:

- Basic gate count. This method uses the gate count of AND, OR, and NOT operators contained in the boolean expressions of a functional unit description.

- Sum-of-cell areas. This measure approximates the active-unit area using sum-of-cell areas where each cell implements one symbol in the schematic diagram. Cell area is available from library data which is obtained from the manufacturer.

- Transistor density product. An approximation of the area can be made by multiplying the number of transistors by the transistor density-coefficient in μm^2/transistor. The transistor density coefficient is derived from averaging the layout area per transistor over all the available library cells or over some comparable design.

For the interconnect area, most estimates are based on the assumption that area is proportional to the number and size of the interconnect units. This can be based on the number of multiplexers, for instance. For determination of the routing area, the number of wires and the number of multiplexer inputs can be used.

4.4.6.5.2 Performance Measures.

Performance measures for synthesized designs have been traditionally based on the clock frequency or number of operations per second. These figures are not always accurate since the actual performance is based on the time necessary to execute a particular description.

An HDL description can contain loops without fixed bounds. IF-THEN-ELSE statements occur where the branches take differing amounts of execution time. In such cases, the execution times are dependent upon the value of input data. For any description, then, the execution time is equal to the number of control steps necessary to execute the description, multiplied by the control step duration (clock period). Techniques used to enhance the performance of a particular technology include (Gajski et. al. 1992):

- increasing the clock frequency,
- identifying critical paths and minimizing control steps for these paths, and
- executing operations in parallel where possible.

4.4.7 Synthesis Advantages and Disadvantages.

Synthesis-based designs deal with higher levels of abstraction. At this level, the design is not tied to a particular implementation technology. Synthesis targets ASIC- and FPGA-based designs. Some of the advantages of synthesis tools over the former design methods include:

- greater flexibility due to higher-level design descriptions;
- designs of great complexity are more easily handled, increasing a designer's efficiency;
- mapping of the design to a specific library is automated;
- optimization is automated;
- due to the level of automation, there is more time to examine design tradeoffs; and
- gate-level errors will be eliminated.

Potential pitfalls of synthesis and synthesis tools include:

- Not all synthesized designs will result in a smaller or faster design when compared to that of a skilled design engineer. A circuit described poorly in an HDL can be inefficient and slower.

- Learning curves can be steep. Not only must proficiency be acquired with the synthesis tool, but also with the particular HDL. Adequate training in the use of high-level synthesis is not always available or utilized.

- High-level synthesis tools often do not provide adequate support for the designer. Tools should make high-level synthesis an integral part of the design process.

- Synthesis tools can contain errors. Design verification is required to ensure that no errors are introduced by the tools.

- Poor design approaches can lead to much more time expended during the gate level design and testing phases. Clearly defined design methodologies are often lacking. For large designs, this can lead to major problems.

- It can be difficult to identify where changes in the HDL need to be made in order to produce an optimal design.

- Designers can have difficulty visualizing the relationship between an HDL and the actual gates which it produces.

Design milestones and decision points are often made at different times for the synthesis design process than those made during a schematic-based design. Committing the design to a particular technology can be put off to later in the design cycle when using synthesis tools. Simulation can be performed at a behavioral level early in the design cycle. Architectural trade-offs and "what if" scenarios can be examined and changes made prior to device development. However, testing and diagnostics need to be designed earlier, so that they are available during simulation runs.

If a design contains functional errors, they cannot be detected by the synthesis tool. A poor design will work poorly whether or not it is implemented with synthesis tools. It will be more difficult to locate timing errors that are caused by poor design. Synthesis tools separate the designer from such details. It must also be remembered that synthesis tools are complex software-based tools that can contain errors. Faulty designs with incorrect logic can occur from software errors in the tool. Tools may also produce incorrect designs based on faulty assumptions residing in the tool. Even changes in HDL coding style can cause problems in the synthesis process.

One other important consideration for locating synthesis tool-induced errors is verification. The designer must compare the behavioral simulation results with the gate-level simulation results. This step can identify many of the errors induced by the synthesis tool, but is very tedious.

4.4.8 Open Issues for Synthesis.

For high-level synthesis, there remain significant issues to be addressed that will enhance the use of synthesis techniques. These include (Gajski and Ramachandran 1994):

- Syntactic variance. Changes in program syntax can produce different results for synthesis. These syntactic variances should be eliminated so that designers unfamiliar with these nuances are still able to produce satisfactory designs.

- Library transparency. Tools should be capable of making use of a large number of available libraries. Without this capability, tools need to be tuned to a specific library, which will make tool maintenance more difficult.

- Interactivity. In order to have control over the design process and get the correct designs, tools need to have provisions for design tailoring, such as interactive interfaces.

- Lower level interfaces. Since designers need to consider the implications of design layout at the lowest levels, tools should provide the designer with interaction for both logic and layout tools in order to enhance the overall design quality.

4.5 ELECTRONIC DESIGN AUTOMATION TOOL STANDARDS.

One of the biggest problems faced by the EDA industry is interoperability among tools from different vendors. While vendors offer large tool suites for ASIC design, no single vendor provides designers with the complete set of tools that is necessary to take a design from start to finish. Thus designers are forced to use tools from multiple vendors. Porting designs from one vendor's tool to another can be a time consuming, expensive, error prone, and possibly an unsuccessful venture.

Multiple interoperability standards efforts by both industry and government have been undertaken to address this issue. Industry efforts include the CAD Framework Initiative (CFI) and the Electronic Design Interchange Format (EDIF) of the Electronic Industries Association (EIA). Government sponsored efforts include the Product Data Exchange Specification (PDES) and the International Electrotechnical Commission (IEC).

A recent entry in this effort is called the EDA Standards Roadmap. It is cosponsored by Sematech (essentially U.S. semiconductor manufacturers), CFI, and the Electronic Design Automation Companies. The concern is that without an interoperability standard, the real issues inherent in deep-submicron high-speed design that now face designers will be overshadowed by software compatibility problems among the various tool vendors. Three areas are targeted for standardization by the EDA Standards Roadmap project. They are

- EDA CAD system interoperability and integration,
- design and data management, and
- technology libraries and models.

At the core of the interoperability problem is the data exchange problem. Design data can have complex and varied representation, depending on its use. Data are used in representing all phases of design such as schematic capture, simulation, placement, and routing. Different vendors also represent data for similar functions in different forms. For schematics, one simple example can illustrate the point. For some, an intersection of two lines on a schematic means that there is an electrical connection at that point. For others, this intersection is simply an open circuit, unless there is a connecting dot at the intersection.

Overcoming interoperability problems can be difficult. In order to do so, in-depth knowledge of the internal data models for the various tools is required. Many tool developers view these data models as proprietary. Hence, efforts aimed at standardization can face considerable difficulty.

Most tool vendors claim commitment to various standards. In reality, tools generally provide the interface for reading in various standard data formats while not providing any choice for data-out formats. Vendors typically do not provide a data-out interface since it is a competitive disadvantage and allows the user to switch to another vendor's tool. Standardization issues will continue to plague the industry unless serious standardization efforts are undertaken by all parties (Donlin 1995).

4.6 FUTURE DESIGN TRENDS FOR HARDWARE.

Significant change is underway in the design automation field relating to how design is performed. One area where fundamental changes are starting to emerge is in design specification. Instead of using an HDL, researchers and manufacturers are seeking to raise the level of abstraction to a graphical representation that isolates the designer from the HDL.

In the future, design automation tools will be used to codesign systems, generating both HDL code and software. System behavior is captured in statecharts and analyzed using an executable specification. This method has already been applied successfully to real applications with results that demonstrate marked improvements in the development cycle.

These tools focus on assisting designers to manage design complexity. Since digital system complexity continues to increase, automation tools will play even larger roles in the future than they do currently. CEs need to be aware of trends that will eventually play a large part in the design of avionic systems. A unified approach in the certification of systems produced by these tools, that will consider the relevant safety issues from both hardware and software perspectives, will be required.

4.6.1 Electronic System Design Automation.

Currently there are vendors that are developing design tools that are yet one more step removed from the actual hardware. These new tools are being referred to as Electronic System Design Automation (ESDA) tools. Some vendors have already announced products, and others will soon follow suit, based on this new design philosophy.

ESDA carries design abstraction to a level beyond HDLs. Instead of being based on RTL, ESDA is based on graphical representation of the design. ESDA tools attempt to eliminate the need to understand HDL syntax and semantics, replacing it with the challenge of learning how to use the ESDA tool.

For ESDA tools, design entry can be performed using a number of means, including:

- state diagrams,
- truth tables,
- block diagrams,
- flow charts,
- data flow diagrams, and
- HDL text entry.

ESDA is a top-down design technique that automates the design by taking designers one step closer to the top of the design chain. ESDA tools most likely represent the next step in design automation.

With ESDA, once the design is entered, the tool then creates the design based on some "predefined underlying hardware architecture" (Saunders and Trivedi 1994).

As with any new tool set, there is some uncertainty as to how useful ESDA tools currently are and whether or not a design will fit the particular mold of the ESDA tool. Additionally, new tools often suffer from a lack of support, have unanticipated learning curves, and are subject to revisions for correcting problems.

4.6.2 Hardware and Software Codesign.

There are three types of hardware/software systems: reactive, transformational, and interactive. Reactive systems constantly monitor their environment and react with sufficient speed to satisfy the requirements. A transformational system reads the input and processes data. After a finite amount of time, the results become available. An interactive system, such as an operating system, interacts with the environment at a self-determined rate.

Reactive systems have specific timing requirements they must meet and are generally deterministic. Input values and timing determine output characteristics. Reliability is often a requirement for such systems.

Most reactive systems are implemented in hardware; however, more are being designed as a mixture of hardware and software. Generally, the hardware and software components are separated and determined early in the design process while their integration takes place late in the design cycle. Reactive systems can be described in a way that is neither hardware- nor software-specific. This permits hardware/software integration decisions to be delayed until later in the design cycle.

There are three ways to view a reactive system: at the system level, the software level, and the hardware level. At the highest level, a system-level model can be constructed that can be used as an executable specification. From a software viewpoint, focus is placed upon the complex, real-time behavior of the system. Hardware designers develop the control logic necessary to implement the specified behavior. The common thread is that each of these three views describes behavior.

Statecharts are tailored to implement the requirements of reactive systems. Statecharts are a graphical language and are an extension of the state machine concept. They have considerable advantages for complex designs, including:

- communicating design functionality clearly and concisely,
- capturing the design's behavior graphically,
- eliminating errors and ambiguities by producing production code in high-level languages directly from the specification,
- implementing design changes more easily than traditional methods, and

- facilitating generation of different hardware/software design combinations, providing a way of repeatedly analyzing a design early in the design cycle.

Statecharts describe behaviors in terms of states and transitions between them. They execute hierarchically while state machines do not. Each state in the statechart can be decomposed into substates. States can exist exclusively or concurrently. When states are concurrent, execution takes place in all substates of a given state.

Statecharts have a rigorous well-defined mathematical behavior; hence it can be demonstrated that this behavior is preserved in the generated code. Multiple architectures and partitions can be constructed easily. One major benefit from this is ease of floorplanning.

A tool based on the statechart method has been developed. It can generate C, Verilog HDL, and VHDL directly from the statechart specification. The C code executes the software portion of the system while the HDL code is used to simulate and later create the ASIC implementation.

Users can experiment easily with different architectures for hardware and software. For a low-end application, the entire model can be generated in C and run as a pure software application. A high-end application may require that the entire model be generated in HDL, synthesized, and programmed into one or more ASICs.

Statecharts have already been applied to a number of applications. One of these applications was for the Iridium Satellite Communications program. This project required the creation of seven ASICs. The VHDL code generated for the ASICs ranged from 18,000 lines for the smallest ASIC, to 300,000 lines for the largest. Over 1,000,000 gates were synthesized for creation of these ASICs. Most notable for this project was the two-fold increase in the overall design cycle time and three-fold increase in productivity (gates/day) when compared to traditional design techniques for systems of lower complexity (Cohen 1995).

5. FABRICATION AND RELIABILITY ISSUES FOR THE PHYSICAL HARDWARE.

5.1 APPLICATION SPECIFIC INTEGRATED CIRCUIT FABRICATION ISSUES.

Long before a design team completes a design, details relating to device fabrication require addressing. In a complex device, rework is costly. Thorough planning and coordination with the device vendor is necessary to bring a complex design to fruition successfully. Fabrication-related issues that need addressing include:

- The vendor should be contacted early in the design cycle to determine whether there are any library issues that need to be addressed. Potential problems can sometimes be revealed by doing a "dry run" with the vendor using a test netlist.

- Vendors may place restrictions on attributes such as signal name length or upper and lower case naming conventions.

- Designers should identify keywords that the vendor has designated. These keywords are to be avoided in the HDL.

- Design size should be estimated beforehand to ensure that it will fit into the target IC.

- Vendor libraries can contain faults and should be verified. Other errors can be introduced in the fabrication of the IC.

- Routing can become more difficult after design hierarchies are flattened,

- Complex designs are more difficult to route. High pin-to-cell ratios can create congestion problems that may need to be corrected during the synthesis phase.

- Gate utilization varies depending on device construction. An increase in the number of metal routing layers increases the number of gates that can be utilized.

- Only foundries can supply the actual values of capacitance and resistance for the interconnects. This information is required when running more accurate simulations.

Other fabrication and foundry issues relating to ASIC testing and test vectors are discussed in section 6.

5.2 ISSUES IN APPLICATION SPECIFIC INTEGRATED CIRCUIT RELIABILITY.

Traditionally, reliability issues for digital ICs focused on physical failure mechanisms. However, for complex ASICs, the term "reliability" has taken on new meaning. Physical failure mechanisms are not the only source of device failure and new techniques for reliability calculations are needed. It cannot be assumed that errors in the design or implementation phase will all be detected and corrected before devices find their way into products.

5.2.1 Reliability and Software Issues.

Currently, complex ASICs are being developed that require hundreds of thousands of lines of high-level code to describe. According to Munson and Ravenel (1993), the pattern of software faults has been shown to be distinctly related to the complexity of the software. The physical barriers that would limit the complexity growth of ICs have not yet been reached. New technologies and techniques are continually under development. The level of required coding is so high that new techniques are being researched and developed to manage the code complexity problem.

Hardware designers, who often are not trained in programming techniques that lead to high-quality code, write the HDL descriptions. On the other hand, some ASIC design is being done by software programmers who generally understand modern programming techniques. Even if this could allay some of the fears relating to the use of proper programming techniques, there remains a larger problem of simulation and verification. A person unfamiliar with hardware cannot address or comprehend what is necessary to perform and interpret these functions. Ideally the best combination is to have designers who are familiar with correct design methods for both hardware and software.

5.2.2 Reliability and Hardware Issues.

IC reliability continues to be an issue of great concern. Increased complexity means higher pin counts. One of the limiting factors for ASIC designs is the number of available I/O pins. Some devices contain over 500 pins and pin counts close to 1,000 will not be uncommon in the near future. There are many potential problem areas just related to packaging, including:

- reliability of the I/O pin connections,
- device mounting,
- heat dissipation for ICs with more than a million transistors,
- temperature sensitivity, and
- electrostatic discharge (ESD) protection.

Since complete testing of complex ICs is not currently practical, device failures under typical operating conditions can go unnoticed. Furthermore, how can it be demonstrated that no failures would occur during "shake and bake" testing if complete testing is not performed? Due to the nature of certain failure modes, stressing environmental parameters may be the only way to induce a particular failure. For instance, a complex state machine controller may have a large fan-out for the clock signal. Suppose that due to thermal cycling, the clock timing changes sufficiently to introduce an operational error in the sequence of the state machine. That error can go unnoticed in test unless the particular test for that condition is executed during thermal cycling. While this may seem trivial for designs containing small state machines, it becomes far more complex when one considers that state machines with thousands of states can be designed and implemented rather easily in the modern design environment.

Reliability is commonly defined as the probability that a device will operate correctly, for a specified amount of time, in a specified environment. Proper design techniques can assist in eliminating some failure mechanisms. ESD, for example, can cause reliability problems. Following adequate design rules can reduce or eliminate reliability problems due to ESD.

5.2.2.1 Failure Mechanisms.

Failure mechanisms for ICs exist in two categories: process anomalies and wear-out mechanisms. Process anomalies are caused by problems in the manufacturing process, resulting in defects such as contamination and ESD damage. Process anomalies are normally detected by "quality" procedures such as visual inspection, burn-in, and thermal cycling. If anomalies escape detection of the quality screening, they can cause acceleration of wear-out mechanisms. Undetected ESD damage, which can weaken thin metalization or insulating oxides, may result in dielectric breakdown or electromigration failures earlier than normal in the life cycle.

Wear-out mechanisms result from certain intrinsic properties of materials. Hence, a reduction in wear-out-related failures requires design-level measures, tight process control, and high-quality materials. There are three primary failure mechanisms that receive much of the attention from reliability engineers. They are latch-up, electromigration, and time-dependent dielectric breakdown.

Accumulation of an electric charge in a gate oxide results in reduced dielectric strength in metal-oxide semiconductor (MOS) capacitors. This reduces the life of the device. Electromigration is a device problem that primarily is of concern for aluminum conductors. Electron flow in the conductors tends to generate an open circuit after a period of time. Factors that can influence this deterioration, and ultimately lead to device failure, include

- design techniques,
- wafer process control,
- metal composition,
- temperature, and
- current density.

Latch-up is caused by various design and device fabrication problems. Causes and effects of latch-up have been discussed in section 3.7.

A common way to represent the reliability life of an IC is by the "bathtub" curve, as seen in figure 44. This curve is the combination of two exponential failure models. Phase 1 typifies the early life reliability of an IC. Production testing, process control, and material inspection are used to identify and remove potential Phase 1 failures.

Failures in Phase 1 decay to a constant rate. For the Phase 2 period of time, this failure rate remains constant and failures occur randomly. Phase 3 is characterized by an exponential rise in the failure rate. Failures such as electromigration and oxide breakdown begin to predominate in Phase 3 (Atmel 1993).

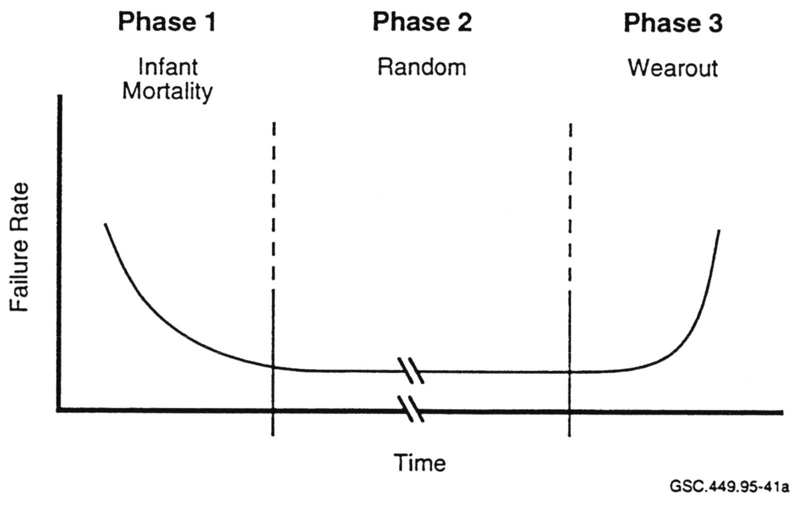


FIGURE 44. RELIABILITY LIFE CURVE

5.2.2.2 Accelerated Testing.

In order to reduce the number of Phase 1 failures in basic CMOS devices, testing of physical failure mechanisms is performed. This involves accelerated life tests and other stress tests. Failure mechanisms that can be found in CMOS devices and their corresponding detection tests are listed in table 10. Typically, a manufacturer will select devices from multiple-wafer lots to ensure that no process variations occurred. A number of tests are then run to ensure that all devices meet some selected quality standard.

High-voltage operating life test operates a device at high temperature and V_{cc}. It is designed to determine the long-term reliability in the operating environment. During this test, devices are operated at 5.5 V (for a V_{cc} of 5.0 V) and 125°C for a duration of 1,000 hours. High-temperature storage tests are performed at 150°C for a duration of 1,000 hours. This test is done without using any bias voltage on the IC pins.

A high-temperature/humidity/bias test is performed to check for device failures in a severe operating environment. Conditions for this test include a temperature of 85°C, relative humidity of 85 percent, and a bias voltage on the pins that alternates between 0 and 5.5 V. This test is run for a duration of 1,000 hours. The package and bonding wires are stressed in this test, and it is also effective in detecting corrosion problems.

TABLE 10. TYPICAL FAILURE MECHANISMS FOR COMPLEMENTARY METAL OXIDE SEMICONDUCTOR DEVICES
(Cypress Semiconductor 1994)

Failure Mechanism	Detection Test
Insulator breakdown (leakage, opens)	High-voltage operating life test
Parameter shifts due to contamination	High-temperature bias
Silicon defects (such as leakage)	High-voltage and guard-banded tests
Metal line opens from electromigration	High-voltage operating life test
Masking and assembly defects	High-temperature storage and high-voltage operating life test
Shorts	Low-temperature, high-voltage operating life test
Stress-induced open metal	Temperature cycling
Open metal from electrolytic corrosion	High-temperature/high-humidity/bias test
Wire bond failure from excessive gold-aluminum interdiffusion	High-voltage operating life test

In order to stress the device, a temperature cycle test is performed. This test cycles the device from -65°C to 150°C for 1,000 cycles. Thermal expansion stress can cause failure of the device package, lead frame, and silicon die. Also, die materials expand and contract at different rates, which can ultimately cause the device to fail.

Two other ASIC reliability tests are the thermal shock test and the pressure pot test. The thermal shock test is similar to the temperature cycle test except that the cycling is done rapidly by using liquids which are set to each of the temperature extremes. This nearly instantaneous temperature change causes higher amounts of stress in the package. The pressure pot test elevates the device temperature to 121°C and the pressure is increased to two atmospheres of saturated steam. This test is a check for bonding pad corrosion and bonding wire integrity (Cypress Semiconductor 1994).

5.2.3 Reliability and Thermal Considerations.

Increased silicon area and higher clock rates have also contributed to the rising difficulties of thermal management. From the design side, every effort should be made to minimize power consumption. Accurate estimates of the final device power dissipation are needed before the device is cast in silicon.

Thermal management has become more difficult as submicron device technology has generated multimillion transistor devices. An IC's junction temperature is a key variable in the equation of the device's long-term reliability. Long-term reliability of semiconductors degrades proportionally with temperature.

Thermal performance of a device is influenced by many factors, including:

- package size,
- package design and construction,
- packaging materials,
- silicon size and thickness, and
- silicon attachment process and materials.

Thermal resistance is the measure of an IC's ability to transfer internally generated heat to the surrounding environment. Heat generated at the junction area of the IC generates an internal temperature that is higher than the ambient temperature. Techniques that lower thermal resistance are essential for ASICs. Thermal resistance is defined as

$$\theta_{JA} = (T_J - T_A) / P$$

where θ_{JA} represents the temperature differential between the ambient environment (T_A) and the die junction (T_J), based on a power dissipation (P) of one watt. This measure may be further broken down into

$$\theta_{JC} = (T_J - T_C) / P$$

and,

$$\theta_{CA} = (T_C - T_A) / P$$

where

θ_{JC}	=	Junction-to-case thermal resistance
θ_{CA}	=	Case-to-ambient thermal resistance
T_J	=	Junction temperature
T_A	=	Ambient temperature
T_C	=	Case (package) temperature
P	=	Device power

Often, ICs are manufactured that require cooling techniques, other than the case-to-air transfer of heat. These techniques include liquid cooling, heat sink mounting to the package, and the miniature package-mounted fan.

6. APPLICATION SPECIFIC INTEGRATED CIRCUIT TEST CONSIDERATIONS.

6.1 UNDERSTANDING TEST.

Traditionally, test methodologies and the approach to testing were the responsibility of test engineers. Design engineering and test engineering were two separated worlds. Test engineers who created effective ad-hoc test methods for ASIC devices containing only a few thousand transistors required as much as 40 percent of the design cycle time (Strickland 1995).

Performing a nonexhaustive functional evaluation of the device is the simplest of all test approaches. The most complete functional test pattern set, referred to as test vectors, can only achieve 60 to 80 percent device fault coverage. The higher the number of device gates, the lower the fault coverage percentage that can be achieved. An exhaustive functional pattern set for devices greater than MSI complexity would require more test time than practical and more resources than are available. For just a moderately complex design, the 80 - 20 rule applies; 80 percent of the effort is spent in trying to obtain the last 20 percent of fault coverage (Strickland 1995).

The device is exercised by the tester using the same pattern set developed during the design process for functional verification through simulation. The simulator pattern sets are reformatted and translated to operate in the manufacturing tester. Certain tester operations, for example those that model tester behavior, are generally included with the simulation pattern set. This is done to ensure that the simulator validation results will be duplicated during device testing. These test pattern sets are not only important in designing the prototype device, but also for ensuring that every device the manufacturer produces works just like the first prototype and is fault free.

A fault is basically a discrepancy between the actual device performance and its expected performance. In general, faults can be introduced into the device in the following ways:

- testing,
- handling,
- manufacturing, and
- design.

Testing induced faults are defined as faults that result from the test method. This form of fault is eliminated by correcting the test system configuration or test conditions.

Handling-induced faults are introduced as a result of mishandling the device. These faults are corrected by establishing and following good quality control procedures and practices.

Manufacturing-induced faults are defects introduced during the silicon fabrication or assembly operation. A silicon fabrication defect is any spot of missing or extra material that may occur in any ASIC layer. Manufacturing faults generally result in an open or short circuit. A short circuit is referred to as a bridging fault and can occur between any two electrical nodes whether they exist on the same or different circuit layers. Bridging faults can vary in resistance from a few ohms to many kilo-ohms and can cause internal leakage currents. Leakage currents can produce either a non-operating device or a device that could fail in service.

Currently, bridging faults account for up to 30 percent of all device faults and will increase as ASIC technology keeps packing more circuits into less space (Malaiya et al. 1992). The physical properties of the defect in the circuit structural layout will determine the fault characteristics. Elimination of these defects through fault analysis is one of the primary goals of testing.

Design-induced faults are found on every device of a given type and introduced into the device during the design, layout, or mask generation steps. The source of a design-induced fault must be identified and corrected in order to implement testing reliably. Design induced faults can be prevented with the use of good design guidelines and design rule checks.

6.1.1 Device Complexity—The Problem.

ASIC device complexity continues to increase at a rapid rate. As technological advances reduce the physical size of the CMOS transistor and improve silicon quality, applications demand more functionality. The result is that more transistors are packed in an ASIC device. This number has increased more than an order of magnitude in the past few years and is expected to continue increasing.

The gap between the complexity of ASIC devices and the ability to ensure their quality and reliability is widening. At the same time, effective testing of these devices is becoming even more critical as the priorities for increased quality and reliability continue to grow (Levitt 1992). Complexity is the primary reason why ASIC testing has not been more thorough. Other reasons include:

- testing has been considered a postdesign process,
- designers are unfamiliar with potential device problems,
- devices have low volume and quick turn around requirements, and
- the foundry wants all test vectors delivered with the design.

The testing methods most critical to an ASIC's success and quality, and requiring the most development effort, are the functional tests. Their goal is to ensure no device faults exist that can cause an ASIC to malfunction; their thoroughness in checking for defects will determine the ASIC quality. Functional testing can be approached from either of two directions, behavioral testing or nonbehavioral testing.

Behavioral testing exercises the ASIC device in the same manner as it is used in the system. Behavioral testing does not necessarily indicate at-speed testing even though clocking rates may be at-speed. Test vectors may not be switching at the ASIC at-speed design rate, because of tester limitations or testing constraints. Behavioral testing is also referred to as operational testing.

Testing all vector pattern combinations does not imply that all possible sequential combinations of test patterns have been used. To test all sequence combinations would be practically impossible to accomplish, because of tester memory limitations and the amount of test time that would be necessary. This limitation in test capability could mask a data pattern or data rate sensitivity problem and thereby allow defective devices to be placed in operation. This could be disastrous for a safety-critical application, such as an aircraft flight control system.

Nonbehavioral testing checks to see that each circuit and interconnection is operational and performs its individually designed function, regardless of how the device is used in the system. Nonbehavioral testing is also referred to as structural testing.

The most common ASIC fault is caused by an internal short to ground or power, or an open circuit, that would produce an external indication that an internal node is stuck-at either a logic 1 or a logic 0 state. This type of fault is referred to as a "stuck-at" fault. It is not the only potential fault condition; however, it does account for a high percentage of the device faults. A stuck-at fault condition has been an effective model used for quality improvement criteria and is the basis for most test methodologies.

Stuck-at fault conditions are simulated in the ASIC design and the results analyzed to determine the fault coverage that can be achieved. The fault simulator determines if a pattern set creates an observable detection condition for a set of potential faults and reports on this activity. Fault simulation is not oriented toward evaluating the functional behavior of the device; but, simulates each logic function to analyze nodal response to a set of vectors.

Fault simulation usually does not perform operations that are essential to functional analysis. There are failure modes associated with ASICs that are not readily identifiable. They can be the result of design errors or subtle internal phenomena that cannot be detected. These include

- clock skew,
- ground bounce, and
- crosstalk.

It is possible that complex ASICs can exhibit data-sensitive behavior due to the cumulative effects of internal currents resulting from peculiar data patterns and application rates. These errors may not be found during testing because of the impracticality of complete pattern testing.

In order to ensure that the fault coverage defined by the simulation process is actually achieved, the manufacturing test equipment and programs must be capable of generating enough test vectors to exercise all the possible logic combinations. Consequently, the results need to be analyzed to verify that no internal node is stuck at a logic 1 or 0 state.

ASIC foundries place the burden of test definition on the designer. From the foundry's perspective, the combination of the large number of ASIC designs and the low production volume of each type will minimize the amount of time and resources they are willing to spend on any single design. Test vectors must be developed by the designer and they must be ready for use in test when the design is delivered to the foundry. Each foundry has a set of specific test equipment with specific and well-defined capabilities. It becomes the designer's responsibility to generate and map the test vectors constrained to the set of capabilities supported by the factory.

Test vector generation is often the task which delays the release of an ASIC design to the manufacturing house or foundry. This is so because the designer may not have the same degree of freedom in generating test vectors as was available to verify the functionality of the design.

Theoretical and experimental studies concluded that to obtain a low defect level in ASIC devices a test pattern set that will test all internal circuits is required. This pattern set is practically impossible to generate manually using brute force techniques (Maston 1994). The less structured the test strategy, the greater the portion of the development time is consumed by testing. Manufacturers are beginning to realize the need to improve ASIC testability and increase quality. As a result, manufacturers are turning to new and nontraditional testing methods. The way an ASIC device is tested can determine whether the development program will succeed or fail.

6.1.2 Design For Testability—The Solution.

As the functional complexity of ASICs and the resulting gate counts continued to increase, manufacturing test became unmanageable. Manufacturing companies began to recognize the need to plan for testability of the device up front with the design cycle. Testability could be ignored during the design phase when designs were implemented in a few thousand gates. The designs were first completed and then turned over to test engineering to work out the test approach and testing details.

As design complexities increased, this ad-hoc approach to testing became futile. Designers realized that if an ASIC design was complex enough to require logic simulation to verify the design and functional accuracy, then it must also be complex enough to include the manufacturing testability requirements of the physical device with the design.

A design for testability (DFT) approach was developed by the industry as a solution to the ASIC testing problem. DFT rapidly became the accepted approach to testing ASIC devices containing over 10,000 gates. High density devices were successfully developed when the circuits were configured with testability incorporated into the design process.

DFT, like any new concept, had to overcome acceptance problems. For example, the worlds of design and test grew up separately therefore the tools, data, and equipment that the design and test groups used did not communicate well. Adopting DFT methodologies within a company meant making a commitment of capital and associated resources necessary to develop a dedicated, trained, and supportive organization. Not every company was willing to make this commitment.

Designing testability into the ASIC adds circuitry and increases the amount of silicon substrate area required. Also, DFT imposes additional constraints in the design and functional verification phases. Some designers argued that incorporating DFT techniques into the ASIC design would therefore increase the cost of the ASIC device. This turned out to be a shortsighted conclusion. The added cost of implementing DFT turned out to be well justified when the savings obtained from enhanced testability and increased reliability were considered.

EDA equipment and silicon vendors responded to the DFT revolution with tools to help engineers integrate test capabilities easily into the ASIC design structure. EDA fault simulation tools were incorporated in the design phase to evaluate the effectiveness of test vectors to detect the existence of a potential fault during design. It was discovered that the number of vector sets required to test for all the potential faults in the device could actually be less than the number of vector sets necessary to perform a reasonably complete functional and operational test of the device.

EDA vendors are producing tools they claim make the adoption of test methodologies practically transparent to the user. The insertion of test logic into the design cycle before logic synthesis illustrates the direction DFT is currently taking (Donlin 1995). These tools provide the design engineer with valuable feedback data from the testing aspects; for example, if a device does not have enough voltage to drive the test pins, the design can be modified with a minimal increase in the design time.

Adopting DFT principles at the beginning of the design process ensures the maximum testability for the minimum effort. From 25 to 40 percent of an ASIC's development time is devoted to test insertion and program generation, and the figure is expected to increase as device complexity increases. DFT techniques can reduce the amount of time spent in test generation. Current test generation tools do create a more testable design but increase total design time. The real savings are realized when the design, manufacturing, testing, and system support costs are all taken into consideration.

The intent of DFT strategy is to achieve the highest fault detection test programs possible in the least amount of time. The higher the fault detection, the fewer faulty ASIC devices end up in the end-user's system (Gruebel 1995). For example, if the fault detection program produced a fault coverage of 100 percent, then it would be possible to detect 100 percent of the defective devices in test. The relationship between the devices with a possible undetectable fault and the device fault coverage level versus process yield is shown in table 11 and illustrated in figure 45.

TABLE 11. PERCENT OF UNDETECTABLE DEFECTIVE DEVICES
(Gruebel 1995)

Process Yield %	Device Fault Coverage %										
	90	91	92	93	94	95	96	97	98	99	100
50	6.70	8.04	5.39	4.74	4.07	3.41	2.73	2.08	1.38	0.89	.00
60	4.98	4.48	4.00	3.51	3.02	2.52	2.02	1.52	1.01	0.51	.00
70	3.50	3.16	2.81	2.47	2.12	1.77	1.42	1.08	0.71	0.38	.00
80	2.21	1.99	1.77	1.55	1.33	1.11	0.89	0.67	0.45	0.22	.00
90	1.05	0.94	0.84	0.73	0.63	0.53	0.42	0.32	0.21	0.11	.00

Parameters are defined as follows:

- Percent of Undetectable Defective Devices is the number of defective devices that tested good divided by the total number of tested good devices expressed as a percentage.
- Device Fault Coverage % is the number of detectable device faults divided by the total number of possible device faults expressed as a percentage.
- Process Yield % is the number of devices that test good divided by the total number of devices tested expressed as a percentage.

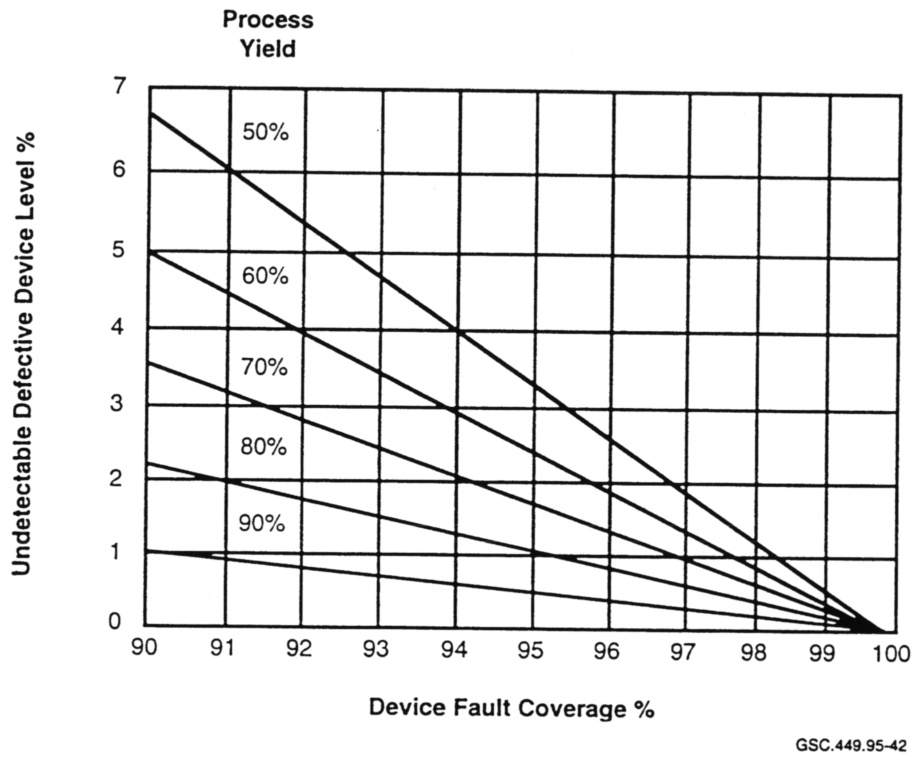


FIGURE 45. PERCENT OF UNDETECTABLE DEFECTIVE DEVICES
(Gruebel 1995)

Table 11 and figure 45 show as an example, for a device fault coverage of 90 percent and a process yield of 90 percent, 1.05 percent of the devices could actually be defective. This means, approximately one ASIC device in each one hundred devices shipped by the manufacturer could have an internal defect. The device defect is undetectable because of the low fault coverage and not from any limitation in testing capability. When the device fault coverage is 100 percent, zero undetectable defective devices will be produced regardless of the process yield value. All defects will be detectable.

The defective device would be installed in the system, pass functional diagnostics, and be placed in operational service. Then, sooner or later, under some particular set of circumstances, the device would fail. The effect of this failure on service is unpredictable but could range anywhere from annoying to catastrophic depending upon the application.

The industry has not established a fault coverage level required for ASIC device testing. Frequently fault coverage levels greater than 95 percent are mentioned as being an acceptable testability goal. Depending on the specific application, this may be too low. For example, for a safety-critical system, a 100 percent fault coverage level may be required to ensure adequate safety.

6.1.3 Test Synthesis—The Support.

With the increasing complexity of ASIC designs and DFT methodologies, the need for a test architecture based on a structured DFT became essential. Tools that would provide automatic enhancement of testability and produce high quality ASIC devices based on a behavioral description of the design became necessary (Halliday and Young 1994). Test synthesis is one test architecture that was developed to meet these requirements and has since been defined as the enabling technology of DFT.

The high-level techniques of test synthesis transform the behavioral description of a design, as written in an HDL, into a structured implementation of data path logic and control logic. It automates the processes of: design analysis, insertion of test structures into the design, and post-insertion design analysis.

Test synthesis is regarded as the indispensable tool required for the successful development of reliable complex ASIC devices. The goal of test synthesis is to create an ASIC that is completely testable under a DFT methodology while meeting device design goals and specifications. Test synthesis improves quality and reduces the time required to produce an ASIC device from design through manufacturing and test.

One key premise of test synthesis is that it allows designers to focus on the fundamental aspects of a design without getting bogged down in low-level implementation of the details. Test synthesis provides the following test-related services:

- automatic design analysis,
- test methodology selection,
- design for testability (rule checking),

- test structure insertion,
- post-insertion design analysis,
- automatic test pattern generation (ATPG),
- automatic scan chain connection, and
- test vector translation.

Test synthesis improves quality and reduces the cost of complex ASIC devices. The quality of the design is improved by implementing the test structures into the design cycle. The implementation of test synthesis tools in the design process makes structural testing easier and reduces the development time due to the following attributes:

- Allows test structures to be incorporated easily into the design.
- Reduces the chance of introducing manual errors.
- Provides a quick recovery from design changes.
- Provides for transportability of test structures.
- Implements test techniques with no detailed knowledge of the test structures.

The high fault coverage attainable with test synthesis dramatically improves the quality of the ASIC device and, as a result, the system. Test synthesis allows test structures to be incorporated automatically into the ASIC design process to improve product quality and reliability and to shorten the design cycle time. Costly errors introduced from previously used manual systems have been virtually eliminated through the automation of these mechanical process steps. Through the use of ATPG and test synthesis tools, the designer is able to implement the necessary test structures and create the foundry test patterns in a couple of days as compared to weeks or even months if the same task were performed manually (Halliday and Young 1994).

Test techniques can be implemented and used without a detailed knowledge of the test structures involved. Additionally, these test structures may now be transported to other similar devices resulting in even shorter development cycle times for new ASIC devices.

Quite often, in the middle of a design cycle, changes are made in the device functional requirements or system interface specifications which reflect the need for a design change. Previously, any change made to the specification during the design process cycle would be difficult and costly to implement. Test synthesis tools provide the capability for improving the implementation and recovery from in-process design changes.

As the complexity and size of the ASIC design increase, test development and tester usage time increase dramatically. Test synthesis tools automate the test development cycle and ATPG tools produce the most efficiently designed set of test vectors that minimize device test time, equipment usage time, and therefore, ASIC cost.

Selecting a DFT methodology is an important step in developing a test strategy. Attention must be given to considerations such as when the methodology is to be used and what environments are to be supported. Design environments include design verification testing, prototype debugging, factory testing, and system integration and test (Halliday and Young 1994). Some requirements to consider in the development of a system test strategy are as follows:

- Available resources and capabilities.
- Ease of use and application.
- Test vectors contained within the system.
- High fault detection.
- Level of fault isolation (substrate area, device, board, and system).
- Execution of tests to be performed on a power-up condition.
- Execution of tests to be performed upon a reset.

The test synthesis tools must be capable of supporting the associated needs and requirements of the various functional groups, such as

- design,
- manufacturing test, and
- system test.

To support the design group, the tools should

- be easy and friendly to use,
- be easily interfaced into the design environment,
- require limited knowledge of test structures,
- require no knowledge of implementation rules,
- provide easy test pattern generation, and
- not change the original design functionality.

To support the manufacturing test group, the tools should synthesize structures that provide

- high device fault coverage,
- isolation of failures within the device, and
- minimum size of test vector set.

To support the system test group, the tools should synthesize structures that provide

- high system fault coverage,
- isolation of failures to a device,
- short test execution cycle time, and
- vertical integration test capability.

Test synthesis may be considered as part of the design cycle; however, it is generally viewed as a back-end design process. Test synthesis is performed after the basic design is completed and verified. Choices made in the beginning of the design cycle will affect test generation and execution. No tools exist today that will weigh these decisions against the test impact until the complete test generation is performed on the completed design.

As the domain of the test synthesis expands and the capabilities improve, choices that are currently made in the beginning of the design will no longer remain as separate design decisions but will become integrated into the operation of the test synthesis tools. The designer will then be able to run what-if scenarios during the basic design and be able to review the end effect on the process. As the EDA industry continues to update their capabilities, a greater degree of compatibility between design and test tools will result.

Testability, which currently accounts for about 10 to 20 percent of the design cost, will continue to be a major cost item and will present an even greater challenge for ASIC designers. With the new mega-gate ASICs and the at-speed testing requirement, testability could become the industry's number one problem and the toughest to solve (Waller[1] 1995).

Combining ASIC test methodologies can create a tester-on-a-chip technology. Testability analysis programs are available which can express the cost of incorporating various test strategies to control and observe test results. From this analysis, testability can be tailored to provide maximum test coverage for minimum cost. Moving the entire test and diagnostic system to the device level drastically simplifies the external test equipment requirements. The ease of use, understandability, and low cost make this an attractive approach to the testability problem.

6.1.4 Test For Safety—The Goal.

Test Synthesis provides the tools to allow ASIC designs to be created by anyone who can define the application. The knowledge of design and test engineering has been programmed into the test synthesis tools creating an expert system for general use. When these development tools have become so automated that operators and application specialists can run the entire ASIC development program, reliability and safety will be totally reliant on the accuracy and capability of the synthesis tools.

These tools must provide upgrade capability in order to remain current with the rapid increase in ASIC complexity and silicon density if they are to maintain their viability. As ASIC technology evolves and circuits become smaller, the ASIC device properties will change, and new problems will develop. Engineers and technical experts who understand the latest technology, must remain part of the development team in order to identify problems and define their solutions.

For example, there are failure modes associated with ASICs that are not readily identifiable since the device functionality cannot be completely simulated. Because of the device complexity and extremely small geometries, certain analog and transmission line phenomena can occur which could generate data sensitivity failures. Often, designers and tools do not account for these effects. Without engineering and technical staff support, defects could easily avoid detection and result in the installation of low quality products in safety-critical environments.

When a problem develops that could affect device quality and reliability, it must first be detectable then accurately diagnosed, evaluated, and corrected by the development team. Technical expertise is required to uncover a potential problem or observe the existence of a disturbing phenomenon that is undetectable using synthesis tools. A fault condition must never be allowed to slip through manufacturing undetected. A device fault discovered only after an operational failure could have catastrophic consequences.

Test equipment, synthesis support tools, and procedures used to test ASIC devices must be carefully evaluated. The test procedures and process must be followed and monitored rigorously, and the results documented and verified. The establishment of a good quality management program is absolutely essential to ensure maximum system safety. ASIC test programs must set a Test for Safety goal. DFT is one of the techniques developed to assist in achieving this goal.

A Quality Test Action Group (QTAG) report stated that the automotive industry is demanding high quality devices. Using the single stuck-at fault test design with a fault coverage of 99 percent is insufficient to detect all of the existing faults. Also, 10 percent of device faults can only be detected using quiescent current testing methods. These faults are primarily in the tri-state buffer circuitry that forms the internal data bus. No one test methodology by itself can guarantee a 100 percent fault coverage (Runyon 1995).

Manufacturers of medical life-support devices, such as pacemakers and implanted defibrillators, state the quality of their products simply cannot be compromised. ASIC devices are used to fill the requirements for smaller size, increased functionality, and reduced power consumption. DFT methods are used to produce high quality ASIC devices. Quiescent current testing methods are used to maximize fault coverage, increase testability, and help achieve the high quality demanded by the medical industry. This test strategy translates directly into higher reliability and greater safety for these life-support devices, which benefits the physicians and patients who use them (Ehlscheid 1995).

Test synthesis tools incorporate test methodologies into the design, but do not address the safety issue. To obtain zero defects, 100 percent ASIC fault coverage is required. If no one test methodology can reasonably provide total fault coverage, then test synthesis tools must have the capability to implement and optimize a combination of test methods in order to obtain the desired results. The test process must be capable of producing high quality fault-free ASIC devices suitable for use in safety-critical applications.

6.1.5 Test Economics—The Cost.

Test economics is a subject that has caused much disagreement among test experts. The conclusion that DFT increases ASIC cost because it adds gates and substrate area to the basic design can undermine the pursuit of quality. Based on such a shortsighted observation, it is easy to conclude that testing costs money as compared to no testing. On the other hand, returned defective products decreases production yield which also costs money.

There are still many designers reluctant to use DFT. Some designers say there are situations that exist where DFT cannot be used for technical and commercial reasons. A company that intends to push current technology for both functionality and performance to get an edge on competition will not want to allocate substrate area to support DFT. They want a cheap product not a quality product. At the same time, this could be considered an unwise market strategy as the devices produced will not be testable and therefore may contain some obscure defect. This defect could lead to a product recall and end up costing considerably more than the cost to implement testability.

Warranty replacement costs could be just a fraction of the total realized cost. The economic issue of testing takes on a whole new meaning when inadequate testing results in lost customers and future sales, payments for lost revenue by the nonperformance of supplied equipment, purchase contract penalties for excessive downtime and consequential loss, and litigation and awards for damages caused by product failure.

Some designers ask why test as a cost is even discussed and consider test as much a part of the design specification as functionality and performance. These same designers feel the word "overhead" should be replaced with "value added" for products incorporating DFT strategies.

The "save money now" viewpoint has also been challenged by the "rule of tens". This rule states that the cost of finding a defective device increases by a factor of ten for each increase in level of assembly. The level of assembly refers to the level where a defect is detected and isolated, either through test diagnostic procedures or an operational failure. The levels of assembly are

- substrate,
- device,
- board,
- system, and
- field installation.

The ten factor may be disputed, but the nonlinearity of cost to level of assembly is realistic. For example, the cost associated with having a device fail in operational service could be astronomically higher than the cost to find the cause of this failure during manufacturing test. In addition, the resulting effects of the failure could have catastrophic consequences.

The total device cost can be determined only after the costs associated with each of the following areas are considered:

- design,
- manufacturing,
- device testing, and
- system support.

Figure 46 shows typical relative costs of an ASIC device compared to the percent of device fault coverage or the reliability goal. This figure shows two sets of curves; one set is based on ASIC development with DFT strategy, the other set without. Each set of curves contain two curves; one to show just the development and test cost, the other total cost. An additional curve shows the combined system and field cost of the device as compared to the percent of fault coverage in the device. The system and field cost is included in the total cost curves which accounts for the high total cost for low levels of fault coverage.

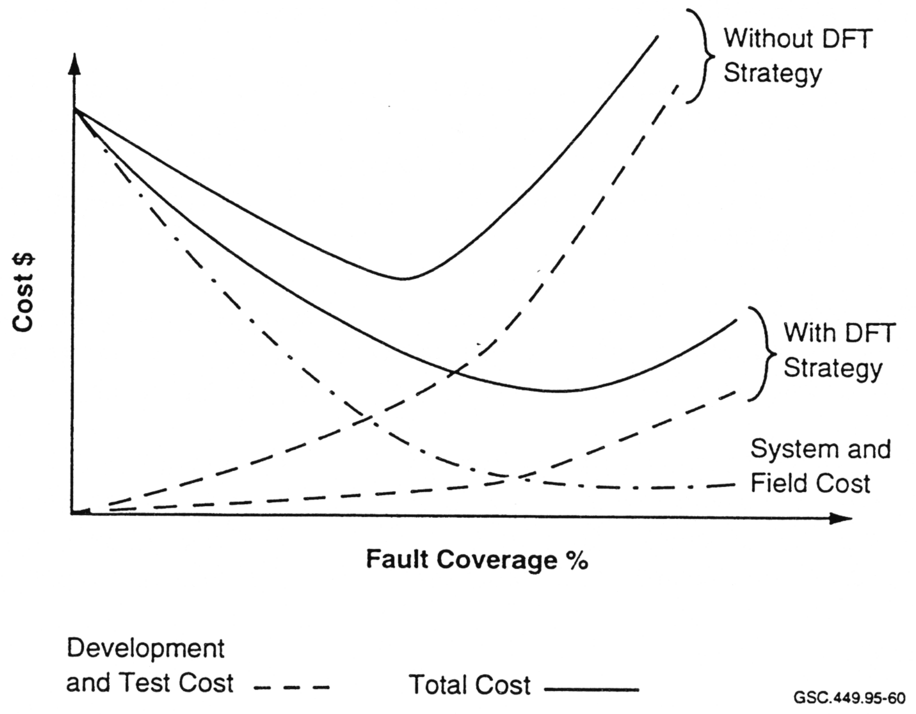


FIGURE 46. ECONOMIC TRADE-OFF FOR TESTABLE DESIGNS (Grueber 1995)

DFT reduces the overall ASIC cost through capabilities that provide:

- elimination of manual test pattern generation,
- faster fault simulation,
- easier access to internal nodes,
- faster device debug time,
- reduced tester cost,
- increased product yield,
- reduced field support,
- increased reliability, and
- decreased liability.

The manual generation of test patterns can take months to complete and may contain many errors because of the manual process. DFT strategies automate this process to reduce time and eliminate manual errors. Large cost savings are realized from these reductions. Savings actually increase with device complexity because test cost, expressed as a percent of total development cost, actually decreases. Figure 47 shows the relative time required for the manual generation of test patterns compared to automatic generation, as provided by DFT strategies. This figure also shows that development time increases with fault coverage. Typical time reductions from months to days for test pattern generation are possible using DFT tools (Gruebel 1995).

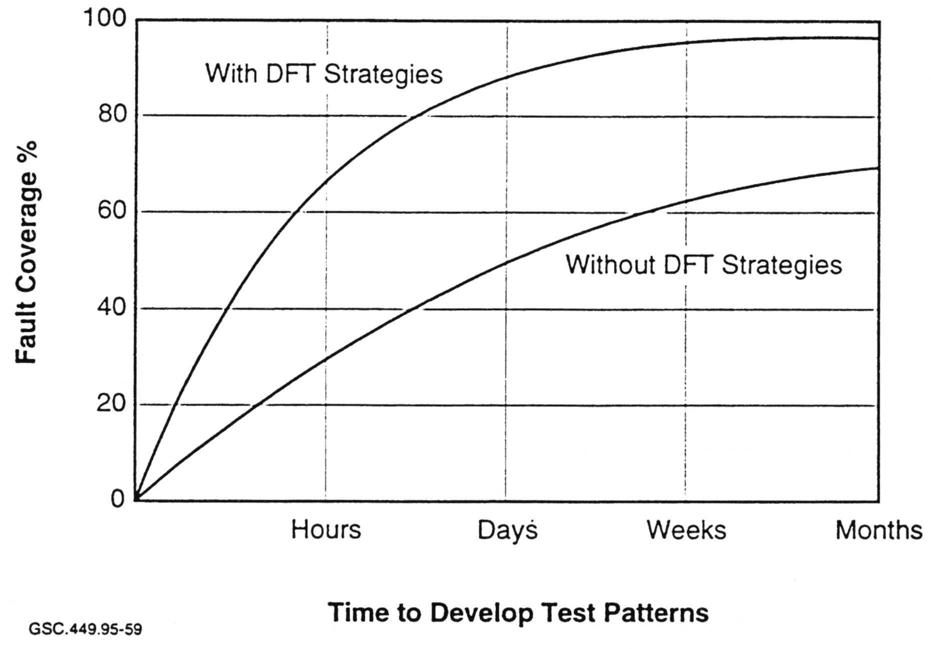


FIGURE 47. TEST PATTERN GENERATION TIME
(Grubel 1995)

The less structured the test methodology, the larger the portion of the development time testing consumes. Testing now requires about 33 percent of the total ASIC development time (Hronik 1994). If nonstructured testing methods are used, the percentage of time that testing will consume increases as ASIC complexity increases. Therefore, the benefits realized from using DFT methodologies increase with ASIC complexity (Ambler et al. 1994).

Testing costs increase as the number of gates and substrate size increase. Figure 48 shows there are cases when the cost of test actually exceed the combined cost of design and manufacturing. This is generally the case when a non-DFT test approach is taken for a complex design.

Figure 48 shows the relative costs of conventional test and scan test programs as compared to device complexity. As complexity increases, the rate of increase in the cost of scan test is less than that for logic design. Since test cost is expressed as a percentage of total cost, this accounts for a decrease in test cost from a corresponding increase in complexity. One particular study showed a test cost reduction of 13 percent was achieved when full-scan testing was implemented for a design containing 17,000 gates (Ambler et al. 1994).

Also, TTM is crucial to the cost-effectiveness and profitability of a design. The analysis of TTM depends upon many variables which include a window of opportunity and the type of product. DFT methods that partition the device into easily testable blocks will simplify test and decrease TTM.

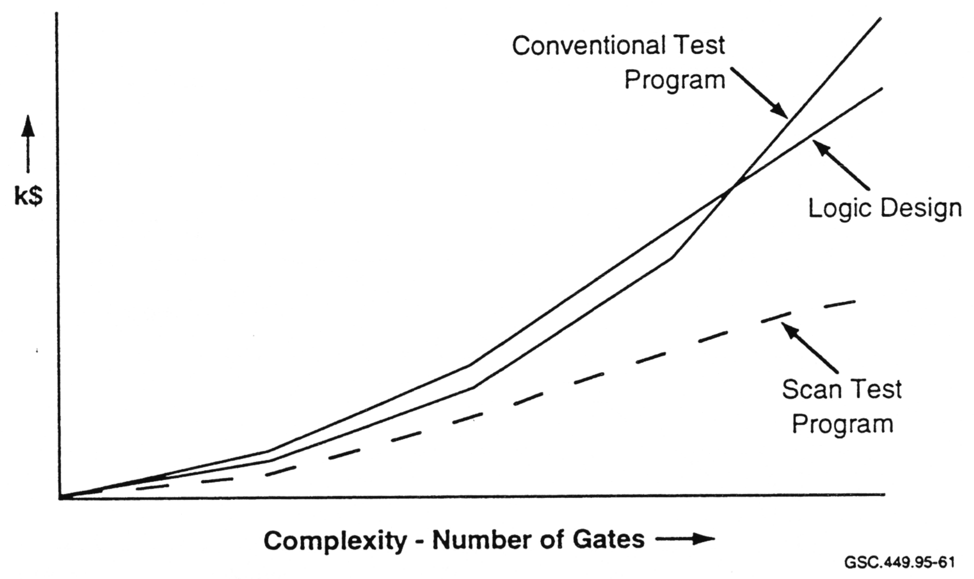


FIGURE 48. COMPLEXITY VERSUS COST
(Ambler et al. 1994)

The amount of time spent in developing an ASIC test program depends on the percent of device fault coverage, which is defined as the percentage of internal circuits that can be controlled and observed. The capability to control the circuit and observe the effect is known as controllability and observability. Figure 49 shows a typical curve which expresses the relationship of test development time to device fault coverage when using DFT structured tools (Levitt 1992). The performance impact of using a particular DFT method may be hard to factor into TTM, but may be crucial to product acceptance in terms of field testability, product quality, and features.

When a product is released to market a few months late, the cost associated from lost revenue could be greater than the cost of having a development overrun (Levitt 1992). If using a particular test technology can get a product to market faster, then the increase in revenue and profits may offset any cost overrun incurred. Figure 50 shows a window of opportunity for a particular product and illustrates the relationship of TTM to the change in revenues when a product is on time or late to market. This graph is based on three market phases:

- Growth phase—sales steadily increase,
- Stagnation phase—sales level off, and
- Decline phase—sales steadily decrease to zero.

The growth phase is the period of greatest profitability. The competition is low and product demand is high. Assuming there are no advantages between the manufacturers' products, the first to market will capture the most revenue. During the growth phase, product pricing is highest because product demand is greater than availability. Profits, therefore will also be high during the growth phase.

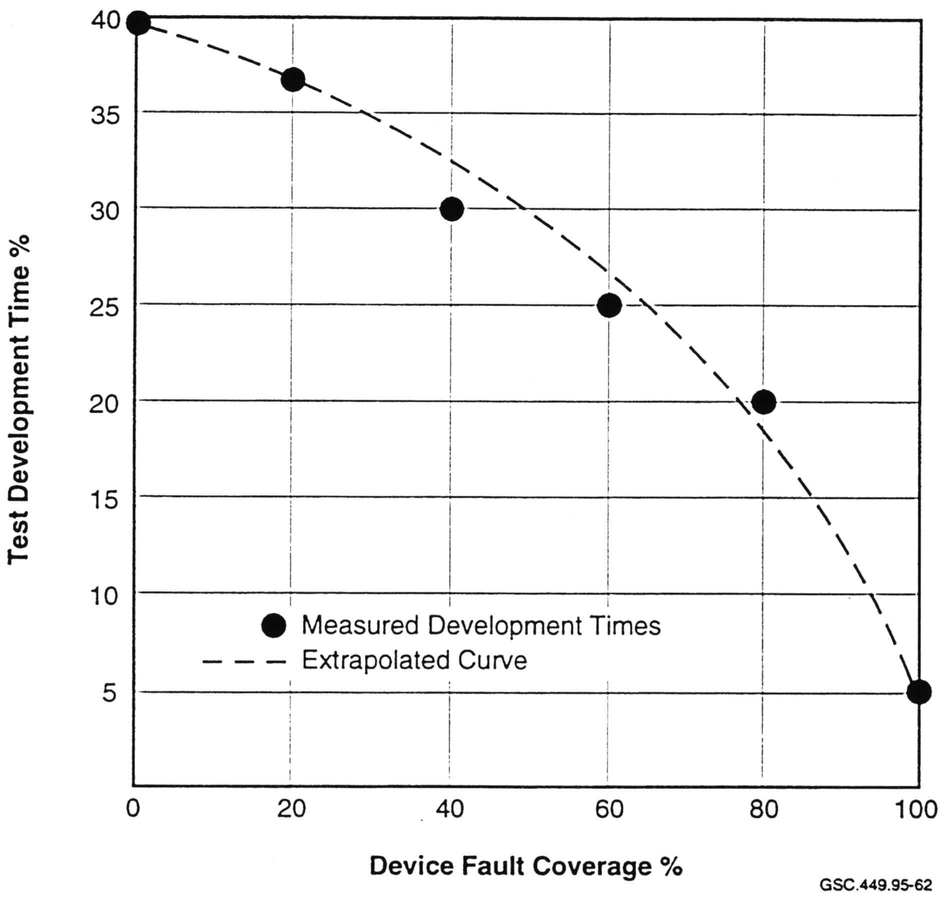


FIGURE 49. TEST DEVELOPMENT TIME VERSUS FAULT COVERAGE
(Levitt 1992)

The stagnation phase begins when product availability becomes greater than product demand. More manufacturers are in the market and the rate of increase in demand is decreasing. Manufacturers attempt to maintain revenue by competitive price reductions to increase sales volume. Each manufacturer is attempting to capture a greater portion of the market with a lower cost product. The product price reduction comes from a reduction in product cost and lower profits.

The decline phase begins when the market demand for the product begins to decrease and the availability increases as a result. Prices have eroded to a bare minimum and manufacturing lowers production volume according to the market demand.

For example, a market has a six-month growth phase, followed by a one-year stagnation and an eight-month decline phase, as shown in figure 50. If the product was late to market by three months, revenue would be reduced by 36 percent. In this case, the delay of one-eighth of the product lifetime will reduce revenues by over one-third.

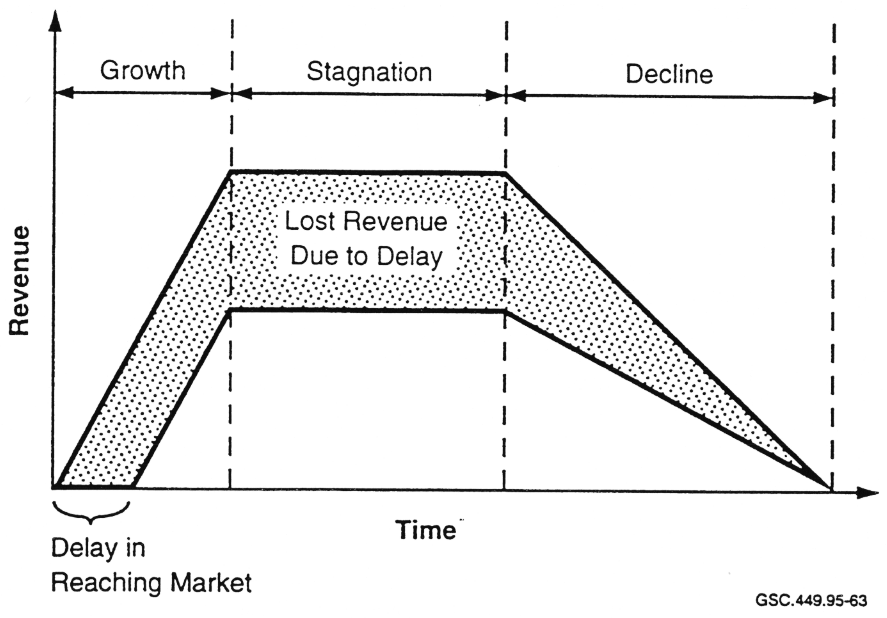


FIGURE 50. REVENUE VERSUS TIME TO MARKET
(Levitt 1994)

TTM is the "profit today" driver to release a product to market as quickly as possible. ASIC customers and users need to be concerned regarding this motivation and therefore must confirm that quality and reliability have not been sacrificed for profits.

The manufacturing device test costs are primarily affected by the number of test vectors needed per device, the number of devices, and the type of test equipment used. The DFT methodology adopted can impact heavily on the number and the type of test vectors to be applied. At times the DFT method can impose restrictions on tester requirements or vice versa. The argument for built-in self-test (BIST) is valid when considering the savings in test equipment cost. The tester is needed to control the ASIC BIST logic only. However, if test time and throughput costs are the most critical factors, which is generally the case in a high volume production facility, then a different DFT method might be selected if a time-consuming exhaustive test would result from using a BIST technique.

There are a number of DFT methodologies available. They each have their own set of advantages and disadvantages. Which one is the most economical and profitable to use cannot be determined based on individual merits alone. The overall problem is much too complex to take a simple judgment approach. The attributes of each methodology must be related to the specific device specification. Considerations must be given to, for example, the overall specification, application, environment considerations, development constraints, and program goals. If the device is to be used in safety-critical applications, the single most important consideration is quality. If no one methodology can provide the required quality and reliability, it may be necessary to use combinations of DFT strategies.

According to the above data regarding cost and reliability, an emphasis on quality is the most economical design goal for all designers to establish and follow. Short-sighted outlooks like "save money now" have short-sighted results.

If testing is thorough, devices that could fail in service will be identified as faulty in test. The level of testing needs to be assessed taking into account a number of factors, not the least of which is the criticality of the system in which the device will operate. Product liability is certainly one cost factor to consider in the overall cost of testing.

The cost of a product liability claim cannot be accurately estimated because of the magnitude of possibilities and resulting consequences that could result from and surround an incident. Taking short-cut approaches to safety like "save money now" can have significant consequences. There is no short-cut to quality. The ASIC must be 100 percent fault-free and developed under strict control of quality management procedures.

6.2 TEST METHODOLOGIES.

6.2.1 Built-in Self-Test.

In order to perform at-speed testing, the test equipment must generally operate at speeds greater than the device under test. Test equipment operating speed requirements necessary to test the submicron and deep-submicron ASIC devices are continually increasing; as a result, the cost of the test equipment is rising rapidly. One approach to solve this problem is to add the necessary test circuits to the ASIC design and provide external device test control I/O capability. BIST is the methodology based on this concept.

BIST essentially allows an ASIC test itself. Creating a device-level BIST architecture enables the ASIC to perform a complete self-diagnostic test for both manufacturing and field system-level testing. This is accomplished by designing a stimulus generator and a response analyzer into the ASIC. During test, the stimulus generator applies a pattern to the device and the response analyzer gathers the results in the form of a long binary data string, compresses the data into a signature, and compares it to the signature of a good device.

The large quantity of circuits used in an ASIC device makes it difficult to achieve a high percentage of test coverage. BIST is one of the most promising DFT methodologies aimed at complete device test coverage (Stroud 1991). The high fault coverage attainable dramatically improves the quality of the ASIC and therefore system reliability. BIST makes it possible to test a device faster, more thoroughly, and for less expense than conventional test methods such as scan. Low-cost test equipment and simple test programs can be used since the bulk of the testing circuitry is integrated on-chip.

BIST encompasses a variety of test architectures that can be designed into a device in order to test for defects in manufacturing or failures in the field (Edirisooriya, Edirisooriya, and Robinson[1] 1993). Most designs use pseudo-random pattern generators to provide test inputs to blocks of internal device logic.

There are three main functional components of a BIST architecture:

- Test Pattern Generator (TPG),
- Output Response Analyzer (ORA), and
- Timing and Control logic.

The TPG is usually designed using a linear feedback shift register (LFSR). The TPG generates the test patterns which are distributed to the circuit under test (CUT) through an internal scan register. The ORA inputs and compacts data and analyzes the resulting signature.

There are several test pattern generation schemes in use:

- exhaustive testing,
- pseudo-random testing, and
- prestored testing.

Exhaustive testing produces all possible test pattern combinations. All detectable single and multiple faults will be detected. Usually a counter, grey-code generator, or a nonlinear feedback shift register (NLFSR) is used for the TPG.

Pseudo-random testing generates a pseudo-random subset of all possible test pattern sets. The fault coverage will depend on the number of test pattern combinations generated. Generally a LFSR is used for the TPG.

Prestored testing uses a test pattern set that was developed during ASIC design and stored in the device ROM. The pattern set is usually very small as compared to the exhaustive pattern set; therefore, it provides limited fault coverage.

To simplify data analysis, many data compaction methods have been developed that reduce the volume of output data, for example:

- signature analysis,
- parity check counting,
- transition counting, and
- ones counting (syndrome generation).

The output data evaluation generally consists of data compaction in which the signature analyzer compacts output sequences of a CUT into a signature of a few bits. Then the BIST controller compares it to a fault-free signature at the end of the test. This comparison determines whether the DUT is fault free. However, due to error information lost through compaction, some error responses may produce the fault free signature, thus causing some faulty circuits to escape detection. This problem is called aliasing or masking (Wu and Ivanov 1995).

The quality of test data compaction is usually assessed by the probability of aliasing, which in turn determines the loss in fault coverage (Kassab, Rajski, and Tyszer 1992). Small signature test registers have relatively high aliasing probabilities and are therefore not well suited to propagate faulty data patterns. Small registers should be merged into larger registers with lower aliasing probabilities (Stroele 1992).

In practice, BIST is used on a portion of a design rather than the whole design. For example, it is used to test large RAM and ROM structures inside the ASIC because they are regular in form and their tests can be reduced to a few program lines that are easily stored and quickly run.

Basic logical structures like memories and multiplexers are simple to test with BIST techniques (Strickland 1995). BIST can be configured to produce a fault coverage of greater than 95 percent for memory blocks using only a few gates.

Random combinational logic blocks are difficult to test with BIST and may require other techniques to be used in conjunction with BIST in order to achieve the desired fault coverage. Random combinational logic testing is usually achieved by building BIST on top of a scan base test architecture, referred to as ScanBIST. With a ScanBIST configuration as shown in figure 51, a scan register chain (SRC) is used to interface The TPG with the CUT and ORA. The ScanBIST TPG is usually based on a LFSR design using pseudo-random pattern generation because of the nonstructured form of random combinational logic.

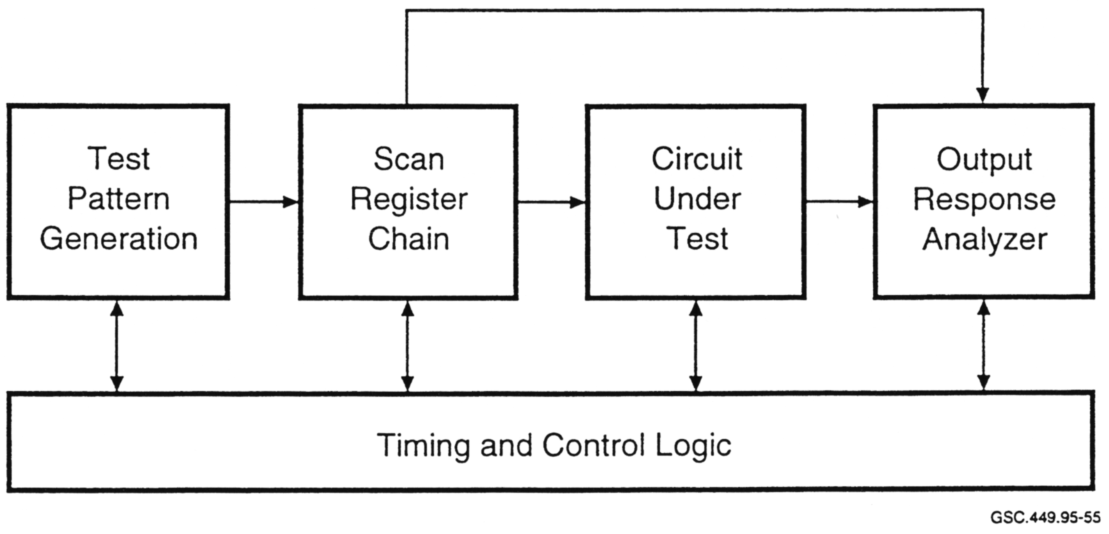


FIGURE 51. SCANBIST BLOCK DIAGRAM

When a BIST method is being considered for testing nonstructured random logic, a thorough and detailed analysis effort is necessary to calculate a good signature. The signature must be stable and repeatable. If the signatures from a good and bad device are identical, or two good devices have different signatures, the test is useless.

One test strategy aimed at improving testability minimizes delay times through the ASIC device by modifying the design of existing internal gates. Instead of adding gates, NOT gates are changed to NOR or NAND gates and control inputs added to existing logic circuits to accept status control signals from the internal BIST generator. Device test points are added to increase the controllability and observability of internal functions to obtain the desired fault coverage.

External automatic test equipment (ATE) functional test programs are used in conjunction with BIST to provide all nonbehavioral testing of the device under test (DUT) such as stuck at 1 or 0. A high level of fault coverage can be achieved from this test strategy and architectural configuration for a relatively low cost as compared to stand-alone methods such as scan test (Motohara and Fujiwara 1984).

BIST can support vertical test integration to extend testing through all the following levels:

- device internal circuit,
- circuit board device,
- circuit board,
- panel or chassis, and
- system.

The device-level self-test architecture enables the device to perform a complete self-test diagnostic with status reporting while operating in a system environment (Agarwal 1995). This integrated test architecture, referred to as the fourth generation methodology, is otherwise known as electronic system test automation (ESTA). The individual self-test units of the system are linked hierarchically so that the entire system is self-testing. ESTA provides a cost-effective and manageable solution for the manufacturing test of ASIC devices via a hierarchial integrated BIST approach. It also offers the potential to perform software design and debugging as well as fault diagnosis and repair at the system level.

Many industry experts believe BIST represents the only viable test strategy for the coming generation of ultra-complex "system-on-a-chip" ASICs. BIST operates at system speed and requires fewer tester interconnection pins than most other automatic test approaches. The internal functional at-speed test capability eliminates the necessity for expensive high-speed external test equipment.

The advantages of BIST testing are as follows:

- Provides high fault coverage.
- Is easy to use and understand (provides conceptual confidence).
- Supports vertical testing (is used at all levels of testing).
- Operates at system speed.
- Requires no complex external test equipment.
- Reduces life-time test costs.
- Enables structured logic testing of memory, etc.

Some of the disadvantages of BIST testing are as follows:

- Requires scan foundation for random logic testing (ScanBIST).
- Requires high silicon overhead (5 to 40 percent).
- Has limited availability of development support tools.
- Has limited ASIC vendor support.

6.2.2 Scan Testing.

Scan test is the most universally effective test architecture used to increase the manufacturing quality of complex synchronous ASIC devices. The goal of scan test is to maximize the device fault coverage by designing total controllability and observability into the ASIC device. Total controllability is having the capability to control the output status of each ASIC circuit on an individual or group selected basis. Total observability is having the means to observe and verify externally the changes in output status of a specific circuit or circuits under control.

Testing is planned from the beginning of the design phase and becomes part of the device specification. Scan test implementation is not considered a separate function but an integral part of the design process. All circuitry necessary to support scan test is integrated with the ASIC design and contained in the device. Any associated increase in internal timing and circuit delays must be factored into the design to ensure that the device meets all the basic ASIC specifications and requirements.

Scan test has been automated with little or no difficulty because the conversion of device storage registers, test vector generation, and test vector application steps are all mechanical operations.

Implementation of scan logic results in an increase in circuitry and substrate size which results in an increase in device cost (Ambler 1994). However, the total cost of the ASIC will be less when considering the overall effect on cost for the following reasons:

- Reduction in test time.
- Reduction in defective devices.
- Reduction of field problems.

Implementation of scan test requires that all storage registers, flip-flops, latches, and counters be designed to facilitate scan operation and provide capability for two operating modes, system and scan test. One of two basic circuit designs are generally used for the scan register, mux-scan design, or level sensitive scan design (LSSD).

Mux-scan is the most widely used design because it requires the addition of only one two-input multiplexer gate before each register in the scan chain, as shown in figure 52. One input is used for scan data, the other for system data. Mux-scan design uses less substrate area than LSSD but increases circuit delay time and therefore degrades performance. The mux-scan data register delay

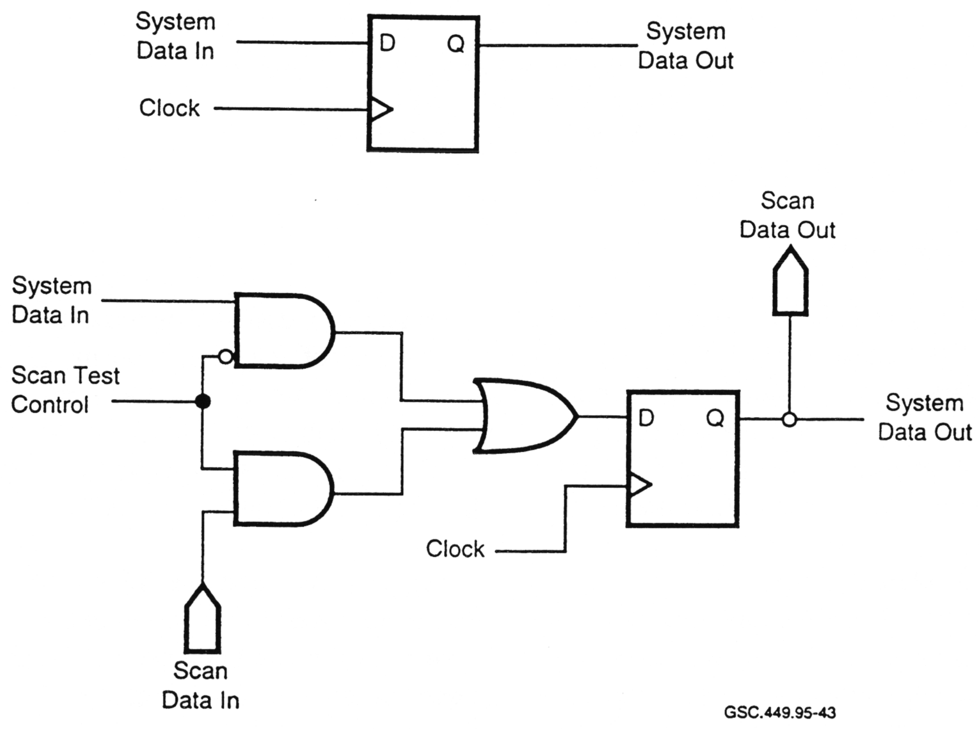


FIGURE 52. MUX-SCAN CIRCUIT CONFIGURATION
(Oakland et al. 1994)

time is longer through the data path than the clock path. Accumulated delay times can cause a shift in the window for clock-to-data setup and hold timing. To correct this problem, scan chain data holding registers are added as necessary to maintain timing within specifications.

LSSD data registers are configured using multiple storage cells as shown in figure 53. The LSSD (level sensitive or transparent latch) circuitry requires a greater amount of substrate area than does mux-scan because of the additional circuitry. LSSD eliminates the delay problem associated with mux-scan and provides greater operational performance (Oakland et al. 1994). LSSD provides unique advantages to the ASIC designer. With any single clock or non-LSSD system scan technique, the designer must ensure that the scan path delay between scan cells exceeds the maximum clock skew variation. With LSSD, the designer needs to be concerned with only system functional paths. Level sensitive, one-clock-at-a-time operation provides a race-free clocking system environment for test. Testing a device for marginal timing designs is not necessary with LSSD. Another advantage of the LSSD technique is that all device logic is testable, including clock gating logic (Oakland et al. 1994).

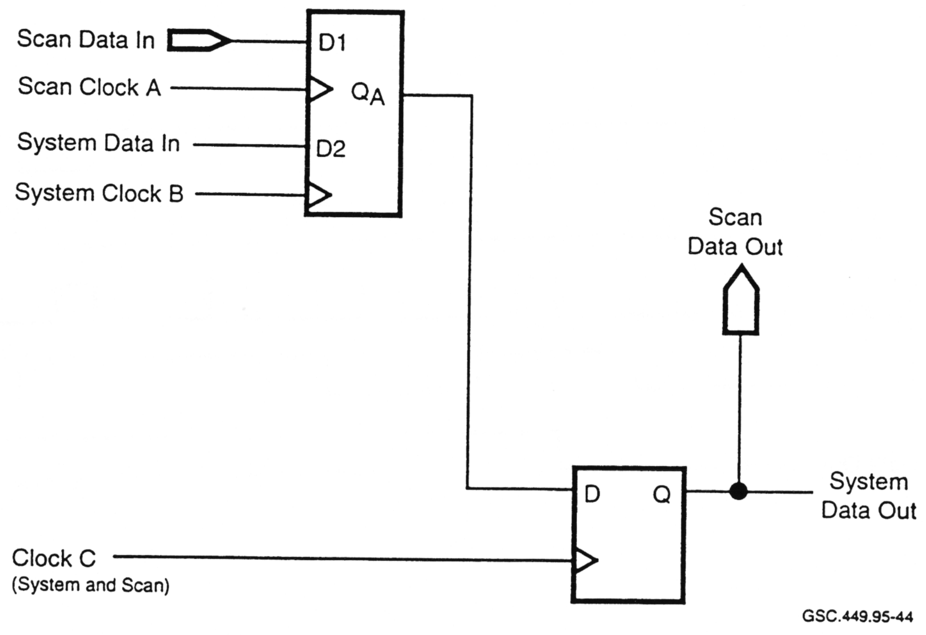


FIGURE 53. LEVEL SENSITIVE SCAN DESIGN SCAN REGISTER CONFIGURATION (Oakland et al. 1994)

Operation of the LSSD register requires a dual phase clocking system. System data out is latched and propagated only if clock A and clock C are consecutively pulsed, as shown in the timing diagram of figure 54. LSSD eliminates the cumulative effects of the data-to-clock timing problem associated with mux-scan design. Scan clock A is turned off during normal operation and system clock B is off during scan operation.

In the test mode of operation, register inputs and outputs are logically reconfigured and interconnected to form a long shift register, or scan chain as shown in figure 55. The shift register scan chain presents a much simpler circuit form to test than the original complex sequential and combinational circuit configuration. Each register now serves as an input and output to its respective interconnecting internal circuits. Test data patterns, or vectors, are clocked into the shift register and later serially shifted through the scan chain and out to the tester for fault check comparison against expected good results.

To test the ASIC device, a scan test control input is activated and an initializing serial pattern of 1s and 0s are serially clocked into the shift register through the scan-in terminal. The ASIC is then switched to the normal operating mode and a tester-generated test vector is applied to the ASIC data input terminals. The test vector is clocked through the input combinational logic by the application of a single clock pulse and the results captured by the scan chain shift register. The device is now placed back in the test mode and the scan chain stored data serially shifted out through the scan-out terminal to the tester. The data pattern is then compared to the expected good data pattern by the tester. This test sequence of operations is repeated for each test vector to ensure no internal ASIC node is "stuck-at" a logic 1 or 0 state.

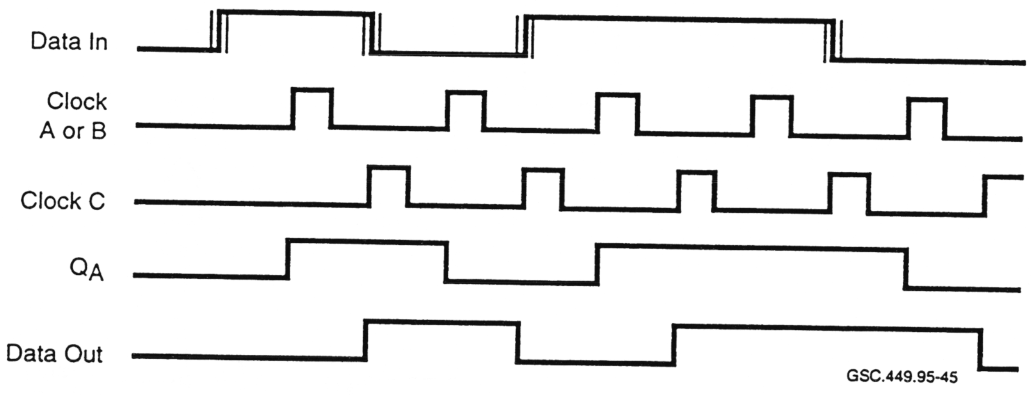


FIGURE 54. LEVEL SENSITIVE SCAN DESIGN SCAN REGISTER TIMING DIAGRAM (Oakland et al. 1994)

The tester hardware must handle large volumes of data. A 50,000-gate ASIC device may well require greater than 3000 scan elements and over 2000 generated test vectors. This will produce more than six million test cycles that the tester must apply to the device in order to test the ASIC thoroughly (Levitt 1992). The tester must be capable of handling these large amounts of data and be equipped to generate the necessary vectors automatically without frequent costly interruptions during the test to pause and reload data.

The test program length is a growing concern to both ATE and ASIC device manufacturers. As the ASIC functional complexity increases, so does the number of test patterns that are required to be generated and tested. This results in the need for more tester capacity and capability. In addition, meeting the at-speed test requirement is not a trivial task and results in even higher tester costs. Exercising the device at system speed requires the test equipment to operate at speeds greater than the device being tested. How to keep the tester cost down and still maintain capability for state of the art technological developments is a major problem of both the ASIC and tester manufacturers.

Improvements in scan test structures have been made which better utilize tester capability and reduce test time without sacrificing fault coverage. Scan-in (SI) and scan-out (SO) operations can be performed concurrently to eliminate half the number of scan operations and the associated test time. During a scan-chain-register shift operation, the present test cycle initialization pattern is simultaneously loaded into the scan chain while the previous cycle's data results are being collected by the tester for fault analysis.

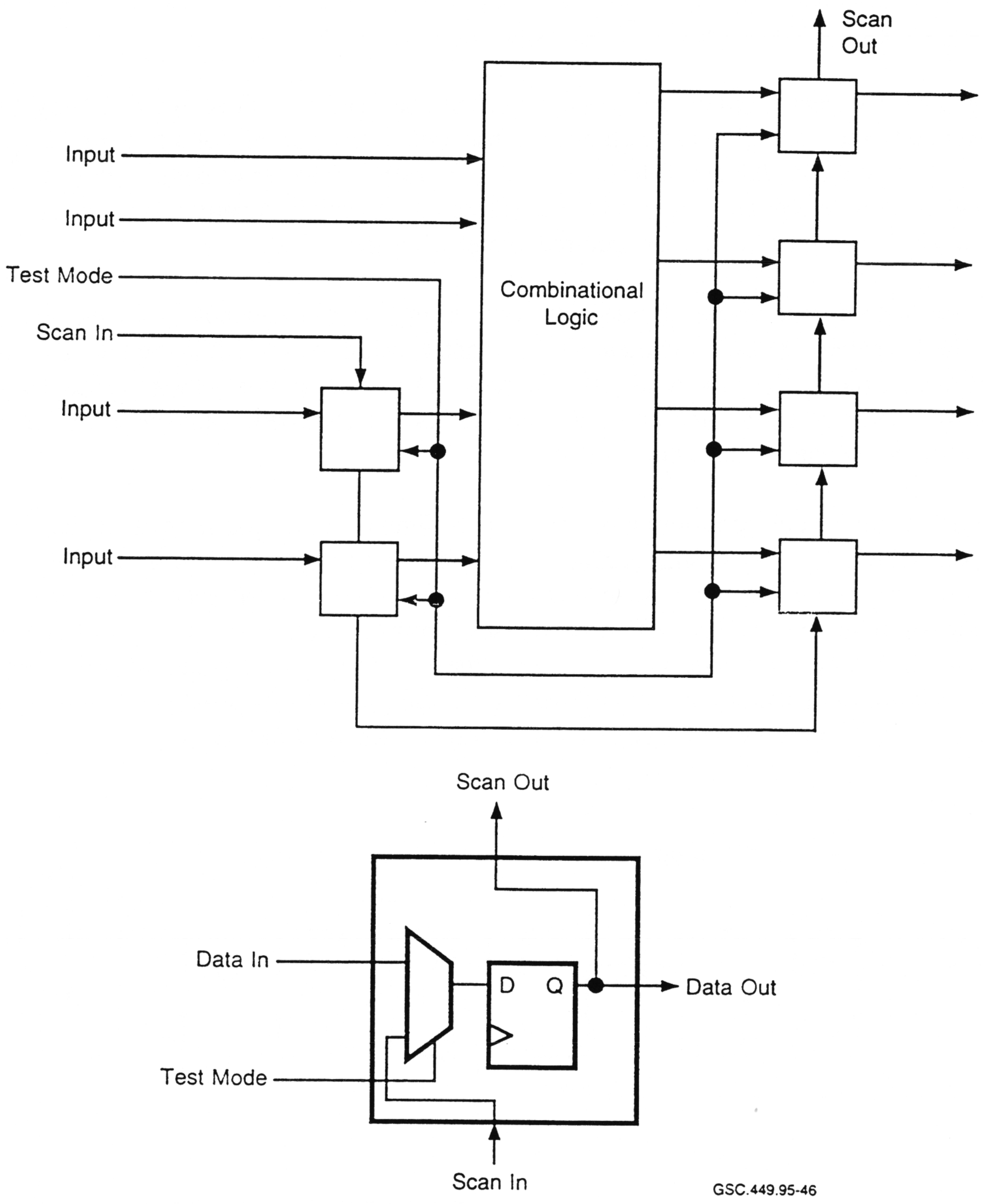


FIGURE 55. SCAN CHAIN CONFIGURATION
(Levitt 1992)

A long scan chain can be broken up into shorter parallel operating scan chains to improve the test structure (Maston 1994). As the number of parallel chains increases, the number of total test vectors and associated test time will decrease. The additional scan paths will increase the number of ASIC device input and output terminals accordingly. In addition, greater tester capability in terms of speed and performance must be provided to accommodate the increase in system throughput. A detailed up-front analysis is required to define the number of parallel scan paths necessary to optimize the overall cost and performance.

Scan control for vector reduction is another technique used to reduce ASIC testing time. This improvement in test structure is based on the observation that portions of a test pattern are repeated many times throughout the test (Morley and Marlett 1991). The tester pattern generator and ASIC scan register can be organized logically and structured according to pattern repetitions to minimize the number of scan shift operations required, thereby lowering testing cost.

The tester pattern generator is divided into two groups, high frequency (HF) and low frequency (LF). The LF pattern generator is maintained on one specific pattern while the HF pattern generator is sequenced through all of the associated patterns. The LF generator is then positioned to the next pattern by a series of shift commands, and the operation is repeated. This cycle continues until all data pattern combinations have been tested.

The scan chain circuit configuration shown in figure 56 is partitioned logically into two corresponding groups of data shift registers, HF group and LF group. A scan control (SC) input is added to select either HF group scan operation, referred to as short-scan operation, or full-scan operation of both the HF and LF shift register groups. Full scan is selected during normal operations and during test operations to update the LF group shift register with the next test data pattern. The ratio of the number of registers in the LF group to the total number of registers, HF and LF groups, determines the reduction in scan time obtained from this technique.

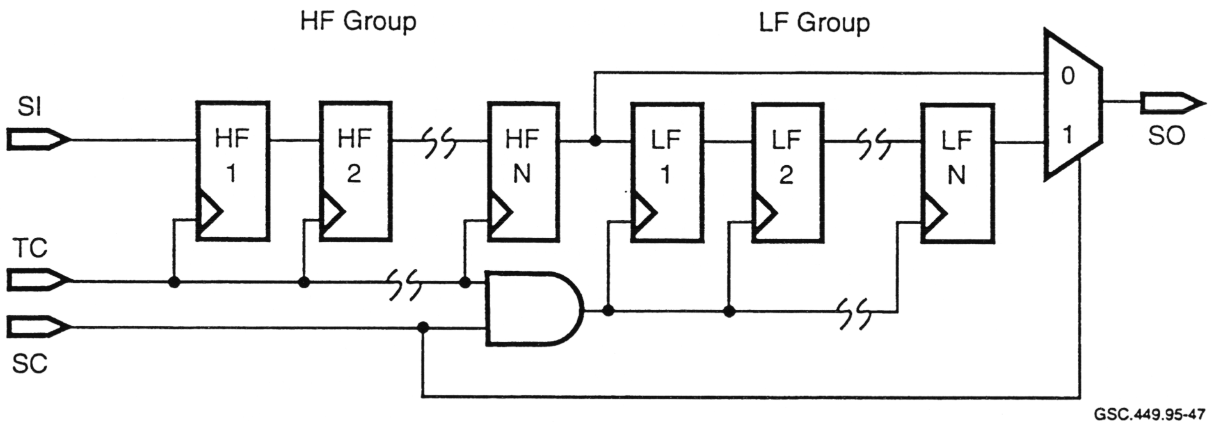


FIGURE 56. SCAN CONTROL CONFIGURATION
(Morley and Marlett 1995)

Operation of the scan control configuration is as follows: when the control signal SC is a logic 0, full-scan operation is selected. Full-scan operation data flow is as follows: from the tester, through the ASIC SI terminal, through the HF group registers, through the LF group registers, through the ASIC SO terminal, and back to the tester. Full-scan operation is selected when the LF group register is to be updated or when the device is not in test.

When control signal SC is a logic 1, short-scan operation is selected. The LF group register test clock (TC) is blocked to hold the current data pattern in the LF group register. Terminal SO is logically switched from the output of the LF group register to the output of the HF group register. Short-scan tester data flow is as follows: from the tester, through ASIC terminal SI, through the HF group, through ASIC terminal SO, and back to the tester. Short-scan operation allows all the associated HF group data patterns to be tested with each specific LF group data pattern.

When other testing methodologies, such as behavioral testing, are used in conjunction with scan testing, additional savings in tester capability and test time can be realized by eliminating test cycles that produce redundant or overlapping test results. The saving in test time from using multiple structured test methods varies and will depend on the specific ASIC design.

The advantages of full-scan testing are as follows:

- Highly structured.
- Good ATPG tools available.
- High fault coverage.
- Good fault isolation.
- Well supported by many ASIC vendors.
- Easily understood.
- Many EDA tools available.
- Easy to implement and use.
- Minimal number of device access terminals required.

The disadvantages of full-scan testing are as follows:

- Large number of test vectors required.
- High test equipment costs (high speed and capacity).
- Works for synchronous logic only.
- Slow; testing done serially.
- Scan register cells larger and slower than nonscan registers.
- High silicon overhead (typically 20 percent).
- Difficult to use for at-speed and stuck-at testing.
- No vertical testing capability.

6.2.3 Partial-Scan Testing.

Some designers are turning to partial-scan techniques because of circuit performance limitations of full scan. Other designers are turning to partial-scan techniques because of previous bad experiences in implementing full-scan techniques. Partial scan presents a trade-off between the ease of testing and the costs associated with full-scan test design.

Less substrate area is required for partial-scan testing because fewer registers are placed in the scan chain. Mux-scan registers use a two-input multiplexer before each scanned register. This configuration, as shown in figure 52, requires the least amount of substrate area but introduces circuit delay time and therefore reduces performance. Registers configured using LSSD, as shown in figure 53, require more substrate area because of the additional registers per cell but there is no performance penalty (Anderson and Allsup 1994).

Logic synthesis tools can analyze the testability of the device and predict the number of scan registers needed to achieve a given test coverage. During this process, recommendations are made for specific registers to be converted to a form that can be scanned, subject to design and specified constraints. If reducing substrate area is a major constraint, then mux-scan registers should be specified for the scan chain. Performance degradation can be minimized by not scanning those registers in critical timing paths.

Scan isolation is a partial scan technique used to test embedded functional blocks of logic, such as a microprocessor core. Scan isolation is based on one of the oldest test strategies ever to be used, divide and conquer. A logic block that has a predefined set of high fault coverage test vectors can be tested easily in isolation from the rest of the design. Scan isolation weaves a scan chain around all the inputs and outputs of the block for serial access from an external tester or the BIST circuitry.

Algorithms have been developed for defining which registers to chain in order to obtain the desired level of fault coverage. Also the ranking level or contribution that each scan chain register has to the overall fault coverage level is defined. Registers can be partitioned to optimize the number of test vectors and the amount of time required for device testing. It is not necessary to include a register in the scan chain if that register is tested by functional behavioral testing.

Sequential circuit test generation tools can be used to combine scan element insertion and ATPG. Starting with a few or no elements in the scan chain design, ATPG is run in a tight iterative loop. If the test generator has trouble in an area of the design, scan registers are inserted automatically into the chain to ease test generation. This process is repeated until the desired fault coverage is obtained or the constraints exhausted.

Other techniques, such as the addition of strategically located internal test points, can be included in the device design to reduce the number of required scan registers and test vectors while maintaining or increasing the fault coverage level.

The advantages of partial scan testing are as follows:

- Allows for user-definable test and DFT goals.
- Supports partitioning by best test method.
- Has less silicon overhead than full scan (1 to 15 percent).
- Has less performance impact than full scan.
- Is easily converted to full scan.

The disadvantages of partial scan testing are as follows:

- Requires more test vector generation time than with full scan.
- Is difficult to translate vectors for tester use.

- Requires completed design to determine fault coverage and silicon requirements.
- Has limited availability of ATPG tools.
- Has limited ASIC vendor support.

6.2.4 Boundary Scan Testing.

Boundary scan test (BST) is based on IEEE Standard 1149.1—Test Access Port and Boundary-Scan Architecture. This standard was founded on a test architecture developed by the Joint Test Action Group (JTAG) for printed circuit board and system-level testing applications (Sherman 1995). The standard provides for replacement of the physical test probes used for functional and interconnect testing to eliminate device I/O signal loading by external test equipment.

Systems designed with components which comply with IEEE 1149.1 can use BST structures for component interconnect test and for sampling component interconnect signals during system test as well as for manufacturing test.

There are three major components to a BST architecture:

- A set of dedicated device terminals for control, clock, and test data, known as the test access port (TAP).

- On-chip control circuitry to direct the test operations, known as the TAP controller.

- A register cell for each device I/O terminal internally connected in a scan chain configuration to form a serial data path, known as the boundary scan register (BSR).

The BST environment controls the device operation through a four- or five-terminal TAP interface to provide complete device access.

The TAP terminals are defined as follows:

- Test Data In - TDI
- Test Data Out - TDO
- Tester Clock - TCK
- Test Mode Select - TMS
- Test Reset - TRST (Optional)

BST provides a method for accessing the inputs and outputs of a device directly without making physical tester-to-device connections, as shown in figure 57. Isolated testability is obtained by the addition of BSR cells placed between each pin of the device and the associated on-chip circuitry. Test data patterns are entered through the TDI terminal and withdrawn serially from the BSR chain at the TDO terminal to perform external interconnect testing or to control the device core logic during internal tests. Resulting data are compared with the expected data in the tester for fault detection of manufacturing defects.

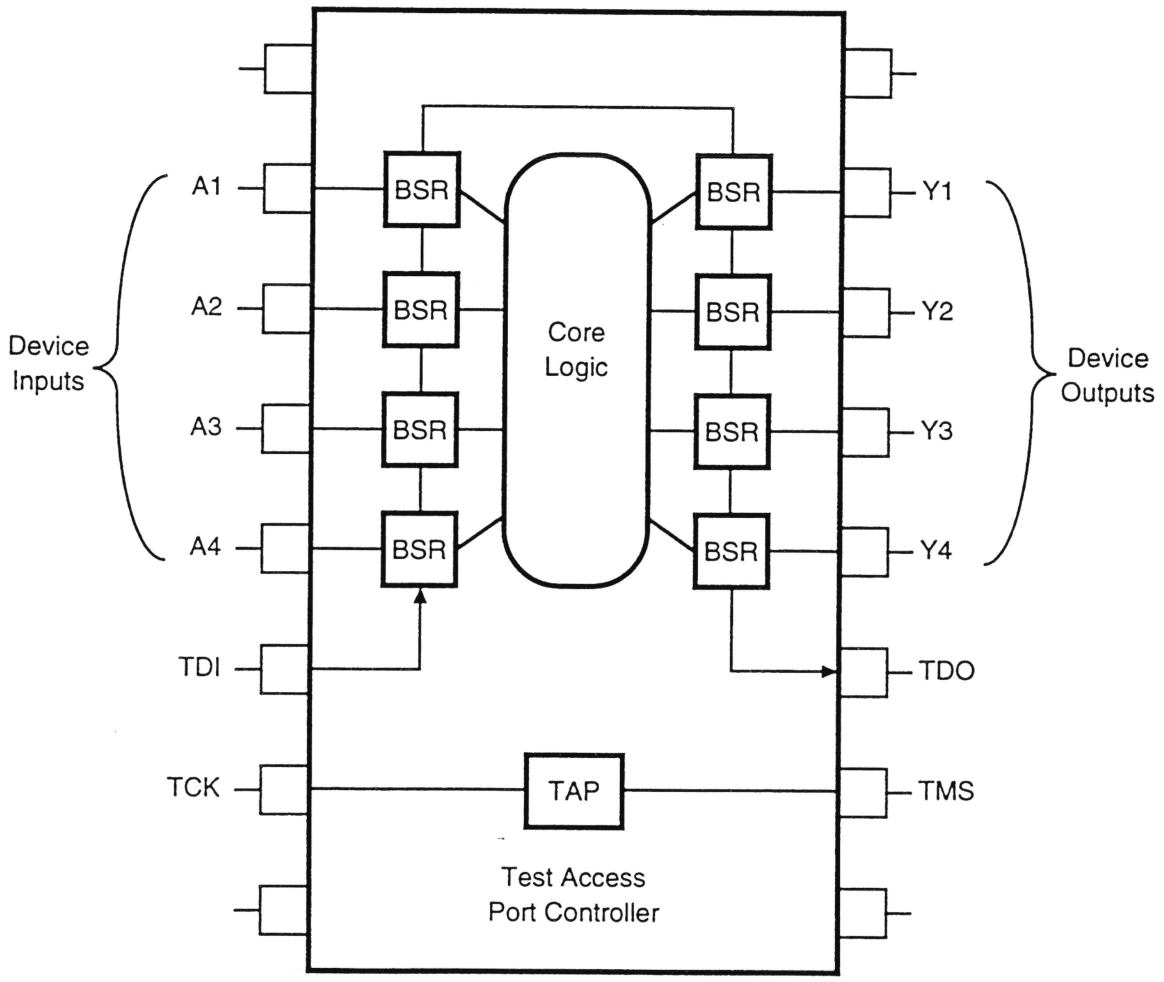


FIGURE 57. BOUNDARY SCAN CONFIGURATION
(Sherman 1995)

Although the primary purpose of the IEEE 1149.1 BST standard is to test the signal interconnections between ICs, the architecture can be expanded and implemented in ASIC devices to verify internal functionality. The TAP controller can be instructed to perform a variety of functions. It can also be used to interface with internal test circuitry, such as BIST to perform diagnostic testing of the ASIC at all test levels, device through system. In addition to the reset capability of the TAP controller, a hardware reset can be performed, if required, by using the optional TRST terminal.

A typical input boundary scan cell configuration is shown in figure 58. Each input cell contains a multiplexer (MUX) that permits data to be entered into the BSR from either the data in terminal or the serial data scan chain, scan in. The BSR allows serial data to be read into or out from the device I/O port. The parallel data register (PDR) maintains data at the inputs and outputs during a scan operation to ensure system signal stability. Each MUX is selected by the TAP instruction register. Through addressing of the TAP instruction register, an external system can input data into the device through the boundary scan chain (Winters 1994).

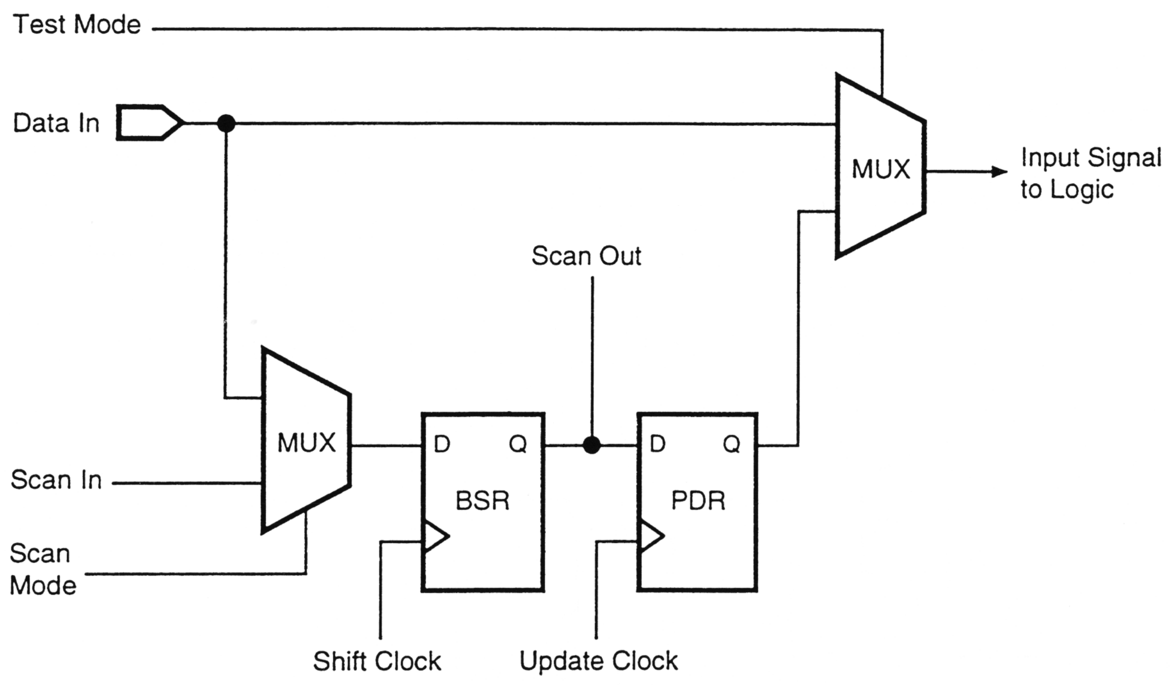


FIGURE 58. BOUNDARY SCAN TEST INPUT CELL CONFIGURATION (Winters 1994)

A typical output boundary scan cell configuration is shown in figure 59. The TAP controller controls the device data output. Signal "scan out" from the boundary scan chain is selected for the device output when the "test mode" signal is a logic 1 and the "output signal from logic" is selected for the device output when the "test mode" signal is a logic 0. The TAP controller directs the boundary scan cells to read data into or out of the device registers and allows the ASIC device to be controlled through the boundary scan chain (Winters 1994).

The IEEE 1149.1 not only defines the device architecture and protocol, but also defines a boundary scan description language (BSDL). Through BSDL, the ATPG tools and simulators are told how a device implements BST (Sherman 1995). A critical step in this process is learning how to develop BSDL files and how to validate their accuracy. Thorough validation is extremely important since the BSDL file forms the basis for testing the device. Inaccurate device descriptions are serious. The BSDL file must describe the design of the device accurately. Even the slightest error can give inaccurate results or damage the device.

Traditional BST design methodologies increase the burden on the designer because the process is still mostly manual. The generation of BST is added to the design after the ASIC core design has been completed. This limitation in design tools requires the designer to consider the effects of the additional scan register delays on the core design. The design verification, analysis, and simulation has to be repeated after implementing BST to ensure the ASIC device still meets the original functional specification (Olen and Hoffer 1994). Also, the designer must verify that the additional substrate area required to support the BST circuitry will not exceed the silicon space available (Maunder and Rodham 1990).

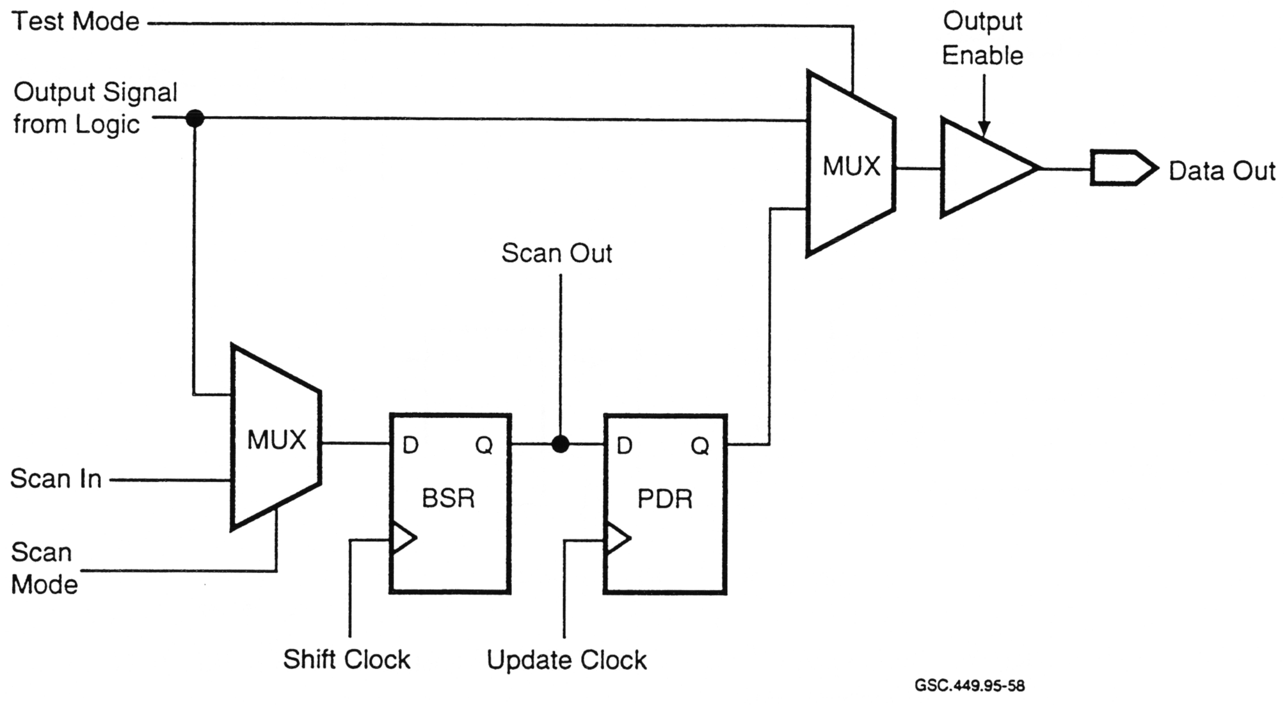


FIGURE 59. BOUNDARY SCAN TEST OUTPUT CELL CONFIGURATION (Winters 1994)

Boundary scan is becoming the most popular diagnostic tool for high performance systems as designers better understand its benefits and implementation (Chenoweth and Muegge 1994). Tools for ATPG and design simulation require the description of the BST implementation. BST architecture has become the de-facto standard interface to on-chip architectures for in-system test and fault diagnosis (Strickland 1995). Manufacturers who have been designing this standard into their devices report no adverse effects from implementing the technology.

BST is capable of performing the most complete system testing possible today in fully working systems. The actual testing is user-friendly once the setup has been completed. BST is a thorough, accurate, precise hands-off testing technique that will not cause degradation of signal integrity or false testing conditions.

Boundary scan testing is most effective when combined with other testing methodologies. In a self-testing ASIC, surrounding boundary scan paths can be used to trigger execution of the self-test, apply the test patterns to the device input pins, and verify the test results. The device will be tested to the full extent of the self-test capability and there will be no reduction in test quality. The slow BST serial data throughput is not a problem because data patterns are only shifted at the beginning and end of the test.

BST by itself is not an at-speed test. BST runs considerably slower than a stand-alone device test because of the need to shift test vectors and response patterns through the scan path. Typically a stand-alone ASIC test program will run one to two orders of magnitude faster than BST (Maunder and Rodham 1990). The low speed of BST makes it impossible to detect delay, timing, and other speed-related faults. In addition, BST is not suitable for testing dynamic circuits which generate transient or charged output signals that decay with time. BST is only effective for detecting static faults, such as stuck-at and shorts. Faults and defects that require at-speed testing to detect will not be found.

Boundary scan is not useful for prototype development. Industry reports indicate half of the ASIC designs produced do not work on the first pass because of inadequate testing and simulation (Fleming 1990). BST does not have the controllability and observability necessary to debug a design. The device must be fully functional including hardware, software, and test methods. A missing portion of the design or an incomplete design will interfere with the test operation and destroy the effectiveness of the BST methods. Also, to develop software fully when the hardware is still undergoing changes during prototype testing is impractical.

The cost of test at all levels can be reduced by developing ASIC test structures based on BST techniques (Oakland et al. 1994). BST structured ASIC devices reduce foundry equipment cost. Only the device scan data, clock, and control pins need to be connected to ATE high speed channels. The scan-based test patterns can be applied with low cost test equipment.

The advantages of BST are as follows:

- Easily understood and implemented.
- Requires low silicon overhead (less than 5 percent).
- Supports high-volume production testing.
- Reduces test preparation and testing time.
- Reduces tester requirements and cost.
- Supported by industry standard IEEE 1149.1.
- Provides field test capability.
- Meets Air Force requirement for fast repair time.
- Easily integrated with other test methods.

The disadvantages of BST are as follows:

- Low availability of automated test synthesis tools.
- No waveform or signal integrity analysis.
- Not effective for prototype development.
- Not an at-speed test.
- No real-time analysis.
- No analog capability.
- No controllability and observability for structural testing.
- Low fault coverage.

6.2.5 CrossCheck™ Testing.

CrossCheck™ is a nonbehavioral, massive embedded observability test methodology developed and patented by CrossCheck Technology Incorporated (Levitt 1992). The entire CrossCheck test structure is completely transparent to the ASIC designer.

CrossCheck adds an overlaying grid of built-in test points to an ASIC's base array as illustrated by figure 60. When the device design is placed and routed, a test point is connected automatically to the output of every gate. Each test point is a simple CMOS transistor that allows the output value of a gate to be transferred to a sense line when a corresponding probe line is selected by an associated probe line driver. The probe lines are implemented in polysilicon and the sense lines are implemented in first layer metal (Lorusso and Fertsch 1991).

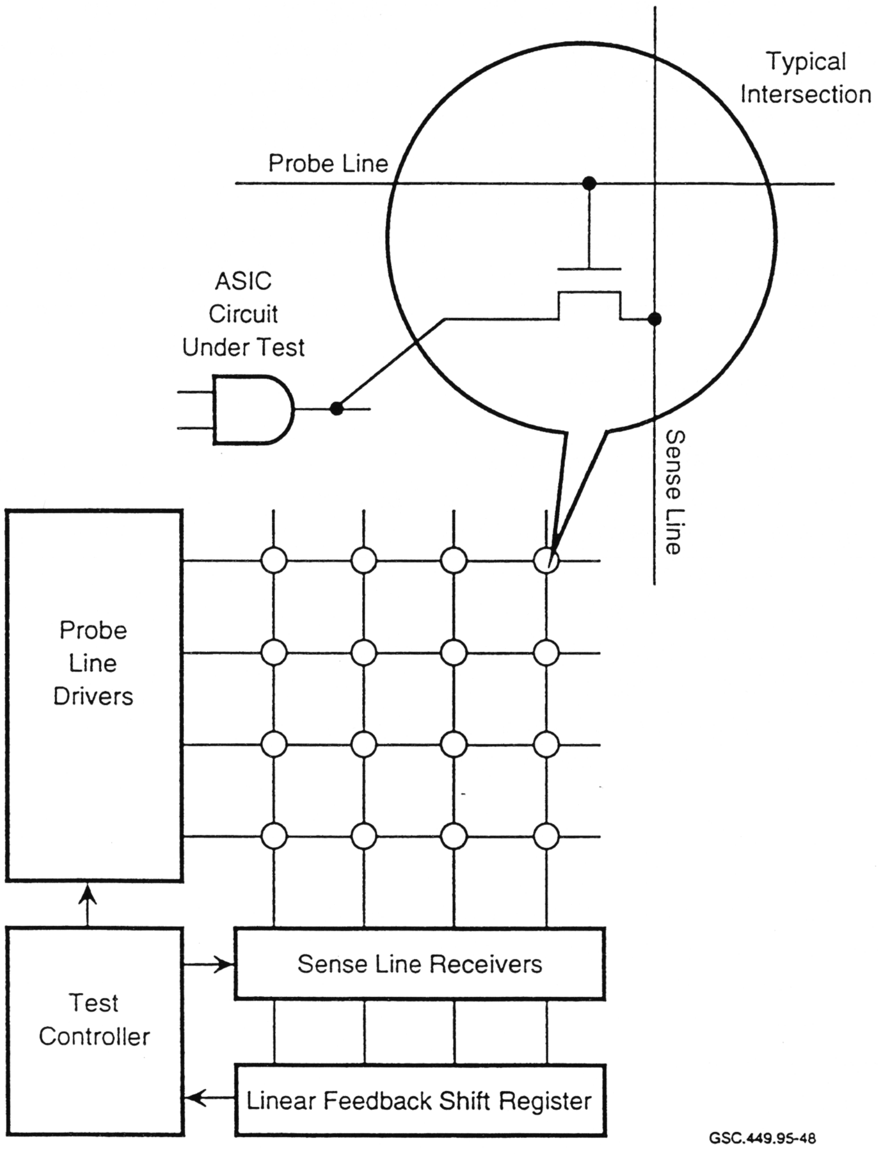


FIGURE 60. CROSSCHECK™ CONFIGURATION
(Levitt 1992)

As shown in figure 60, the sense lines connect to an analog sense amplifier which conducts a voltage level analysis to determine if fabrication defects are present. The output of the sense amplifier is connected to an LFSR which compresses the test data into a signature. The signature is then shifted out for comparison with the known good signature for operational verification and fault analysis. All probe lines with test points are sequentially activated for each vector in a test before the next test vector is applied to the ASIC device input terminals.

The Test Controller performs four major functions:

- interface to the TAP,
- vector generation,
- signature generation, and
- on-chip self-test.

Access to the on-chip self-test unit is through a four- or five-pin TAP. The TAP is used to interface with external ATE and is compatible with the IEEE Standard 1149.1 described in section 6.2.4.

Controllability of internal nodes is achieved through CrossCheck software that provides fault simulation, ATPG, and statistical analysis capability. ATPG is accomplished without modification to the designer's circuit configuration network listing (netlist).

NEC provides CrossCheck arrays for ASIC designs containing 200,000 to 500,000 gates with a significant percentage of asynchronous circuitry (Wilson and Lammers 1995).

The advantages of CrossCheck testing are as follows:

- Provides extremely high fault coverage.
- Works for both synchronous and asynchronous circuits.
- Requires no design-in effort.
- Supported by development and simulation tools.
- Produces little or no performance impact.
- Able to test for nonstuck-at faults.
- Provides fast simulation and fault diagnosis.
- Supports vertical testing through BST.

The disadvantages of CrossCheck testing are as follows:

- Slow (not at speed).
- Requires moderate silicon support (typically 15 percent).
- Available only from licensed vendors.
- Limited support (patented process).

6.2.6 I_{DDQ} Testing.

I_{DDQ} is the IEEE symbol for quiescent power supply current in MOS and CMOS circuits. I_{DDQ} testing methodology is a massive observability technique that will not only detect failures but will also identify leakage currents that could cause the ASIC device to fail in service. I_{DDQ} testing detects device faults by measuring the quiescent steady state power supply current flowing through the device. This quiescent current measurement method of testing is based on two fundamental properties of CMOS gates used in most ASIC designs (Levitt 1992):

- A CMOS gate draws very little current (pico-amps) when quiescent or in the static state (nonswitching).

- Most CMOS gate defects will cause a large current to flow in the quiescent state.

Leakage faults cannot be detected by conventional logic testing methods regardless of how many or how the test vectors are developed. I_{DDQ} testing is being used successfully in production test programs to detect these low level leakage currents in ASIC devices. Since these leakage currents indicate internal defects, which can cause failure of the device, any device not passing this test will be rejected immediately.

Automotive industry studies show that about 10 percent of the faults are detectable only using I_{DDQ} testing. These faults are mainly associated with the tri-state buffers interfacing with the internal data bus. These faults would be classified as undetectable without I_{DDQ} testing (Runyon 1995).

I_{DDQ} testing of ASICs is vital to medical life-support device manufactures since it increases test coverage and helps ensure the highest quality product demanded by the medical market. CMOS ASICs are being used in devices, such as pacemakers, to reduce power and size. These ASIC devices are designed for mixed-signal operation and contain analog and digital circuitry. Implementing I_{DDQ} testing increases the fault coverage without adding test points and reduces test time. I_{DDQ} testing detects the failure mode encountered most commonly in ASICs developed for medical life support products. More CMOS circuits have been found to fail with their output voltage at a value between the supply voltage and ground than have been found to fail with their output voltage stuck at a high or low value. Nonstuck-at faults are very difficult if not impossible to detect without using I_{DDQ} testing methods (Ehlscheid 1995).

To use I_{DDQ} methodology reliably in a production test program, the device must be static or remain in a particular logic state long enough to make a precision current measurement. The current through CMOS circuits is in the area of a few nano-amperes; however, with a bridging fault, the quiescent current will increase, generally by several orders of magnitude. Therefore, if an increase in I_{DDQ} is observed, an internal circuit is considered to have a fault.

In production testing, typically only 20 vector patterns are chosen for I_{DDQ} testing. This choice is a trade-off between test coverage and test time. Halting a functional test to make a precise current measurement at thousands of states would result in unacceptable test time for a manufacturing environment.

Current can be monitored with either external circuitry interfaced to the device terminals or by current sensors built into the device. Because of the vast number of ASIC device circuits, multiple V_{DD} terminals are often used to distribute power to the various functional areas of the substrate. This allows fault localization to a specific substrate area and provides a greater degree of accuracy in discriminating between a fault and a no-fault I_{DD} measurement.

Accurate off-chip I_{DD} measurements are not always easy to acquire using existing test equipment, because the resolution of these low current measurements is limited. One problem is that I/O drivers consume most of the current causing fluctuations in the power distribution system that overshadow most abnormalities and defects in the functional circuitry. Low current measurement resolution is critical in detecting defects which cause only small abnormal current changes, such as those caused by floating gates. On-chip I_{DD} measurements can be made only at a very slow rate. Even with tester modifications, reliable device low current measurements cannot be obtained using a large number of input vectors (Nigh 1992).

Accurate on-chip I_{DD} measurements can be acquired using on-chip built-in current sensors to detect abnormal power bus currents. This method of measurement greatly increases the effectiveness and efficiency of I_{DDQ} current testing. It solves the low current resolution problem since internal functional logic circuits can be connected to separate on-chip sensors. On-chip current measurement resolution can detect much smaller values of abnormal current and allow I_{DDQ} testing to be operated at a much higher rate of speed than is possible with off-chip measurement techniques.

I_{DDQ} testing has the greatest impact on quality and cost when performed at the substrate level. Immediate process test feedback eliminates the costly assembly of defective devices. In addition to substrate level testing, the I_{DDQ} tests should be repeated after the burn-in cycle which activates hidden defects. These tests are most effective with V_{DD} operating voltage set to the maximum specified value for worst case leakage testing (McEuen 1992).

There are various causes of leakage current. Gate oxide shorts are the most frequently referenced faults, but others do exist. Generally, fault causes can be divided into four separate categories:

- testing-induced,
- handling-induced,
- manufacturing-induced, and
- design-induced.

Testing-induced leakage is described as any leakage that results from the test method. This form of leakage is corrected by altering the test conditions. Sources of testing-induced faults are as follows:

- Excessive tester loading of the device I/O terminals.
- Defective tester interface circuit board.
- Insufficient measurement settling times.
- Excessive parasitic capacitance.
- Tester interface connections and wiring problems.

Handling-induced leakage is the result of mishandling the device. Damage to the device can be caused by ESD or mechanical stress.

Manufacturing-induced leakage is defined as any leakage occurring as a result of the silicon fabrication or assembly operation. Elimination of this type of leakage is one of the primary goals of I_{DDQ} testing.

Examples of manufacturing-induced leakages are as follows:

- Gate oxide shorts.
- Polysilicon and metal bridges.
- Low transistor thresholds.
- PN junction leakage.
- Transistor punch through.
- Mobile ionic contamination.
- Stacking faults.
- Particulates.
- Inner-layer oxide shorts.

Design-induced leakage is found on every device of a given type and introduced into the device during the design, layout, or mask generation steps. The source of this leakage must be identified and corrected in order to implement I_{DDQ} testing reliably. This form of leakage can be prevented with the use of good design guidelines and design rule checks.

Examples of design induced leakages are as follows:

- Mask defects.
- Floating gates.
- Incorrect alignment or registration.
- Parasitic devices.
- Internal oscillators.
- Internal resistors.
- Analog circuitry.
- Sense amplifiers.
- Redundant memory.

The cause of all failures cannot be determined. A failure could be the result of a design defect, a design problem or a manufacturing problem. Failures falling into all three of these categories are found in the early testing phase of new devices (McEuen 1992).

As well as being used in manufacturing test, I_{DDQ} testing has been and will remain a useful and effective tool in device failure analysis work. This testing method has design principles that inherently provide high defect coverage, as well as diagnosis capability and physical fault localization (Sylvestri and Quimet 1995).

Dynamic I_{DDQ} testing can be performed on ASIC devices to increase the range of testability. A failing device may not indicate a faulty current level in a quiescent or steady-state condition even though the device design is fully static. The dynamic current signature of an ASIC device is stable and repeatable as shown in figure 61. The I_{DDQ} test is therefore reliable during dynamic testing. Tests are run at speeds low enough to obtain stable current readings and high enough to approach operational speed.

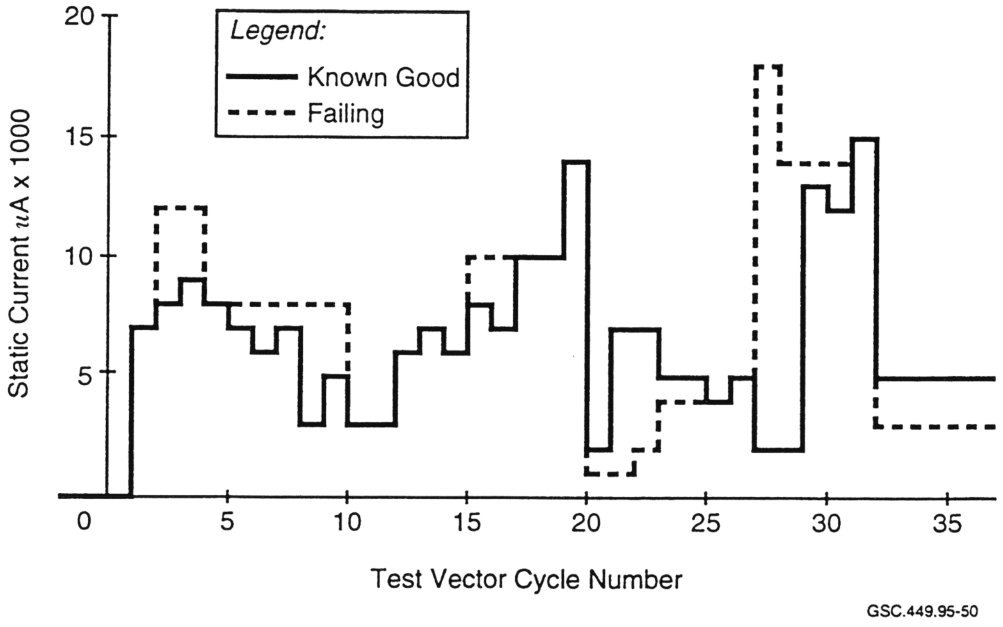


FIGURE 61. I_{DDQ} SIGNATURE PATTERNS
(Sylvestri and Quimet 1995)

Bridging faults in CMOS circuits can cause a circuit node to be connected to both V_{DD} and ground, which could result in a faulty internal logic value. Figure 62 shows an example of this type of fault at an output node (Keating and Meyer 1992). When Q1 is on, Q2 should be off resulting in no static I_{DD} flow. With a leakage resistance in Q2, there will be a measurable I_{DD} flowing through the node to ground. If the resistance value is high, the device will pass all behavioral tests, be shipped as a good device and placed in operation only to fail in service at some later time.

I_{DDQ} testing has been used successfully to augment traditional functional and stuck-at fault testing. It is increasingly being used to improve ASIC quality but is not considered a replacement for an at-speed functional test (Knack 1993).

For high reliability devices, as required for safety-critical applications, the test program should include I_{DDQ} testing that will toggle all logic nodes in the device. Experimental data show that defect detection increased between 60 and 80 percent when I_{DDQ} testing was implemented in the test program (Hawkins et al. 1992).

Technical issues of concern to both ASIC device and test equipment manufacturers regarding I_{DDQ} testing include mixed-signal and at-speed testing. No test equipment today will do a good test of mixed analog and digital signals. Also, it is becoming extremely difficult to provide test equipment capability that can keep up with the advancements in ASIC technology development.

Market demand is stimulating improvements and innovative solutions in hardware and software tools to support I_{DDQ} testing. Product maturity is expected in the next two to four years (Hawkins et al. 1995).

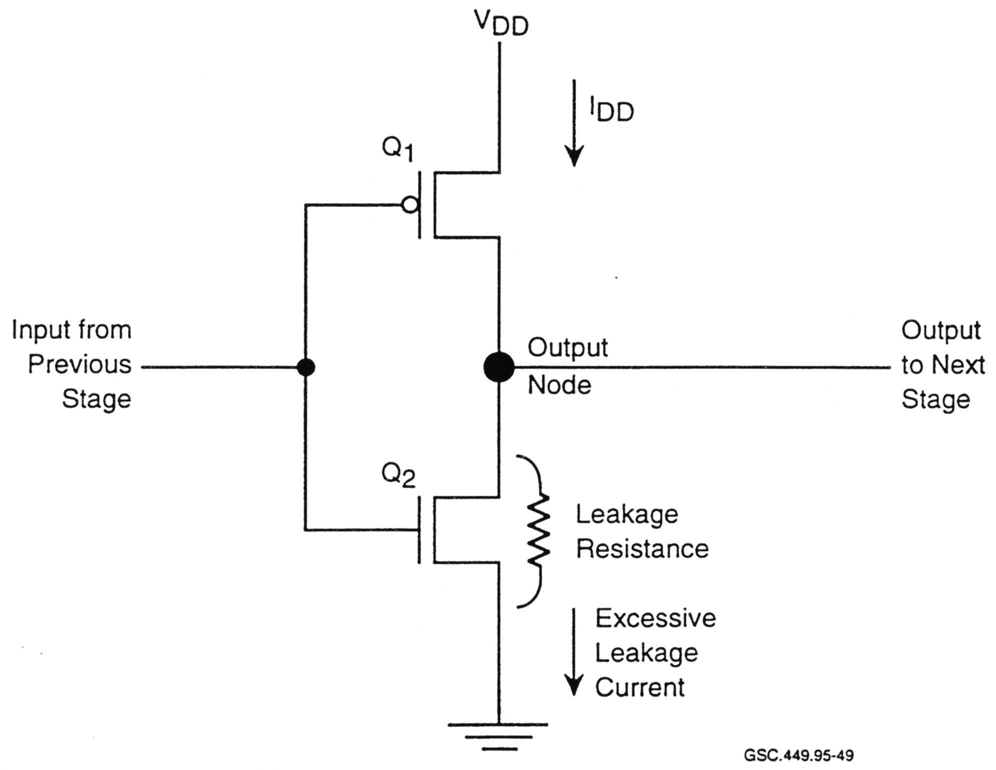


FIGURE 62. BRIDGING FAULT

Current hardware advances have pushed production test rates up to 1 MHz and measurement resolution down to 200 nA. New challenges confronting I_{DDQ} testing include deep-submicron ASICs with up to 100 million transistors, frequencies greater than 100 MHz, low power supply voltages, and hundreds of power supply pins (Malayia et al. 1992).

The advantages of I_{DDQ} testing are as follows:

- Detects reliability problems.
- Provides high fault coverage.
- Tests for nonstuck-at faults.
- Requires little or no silicon support.

- Complements other test methodologies.
- Greatly improves testing strategy.
- Reduces the number of circuits and test vectors required.
- Is capable of high speed operation.

The disadvantages of I_{DDQ} testing are as follows:

- Requires isolation of circuits that draw static current; for example, static memory, sense amplifiers, and tri-state output drivers.

- Has low to moderate speed operation (100 kHz to 1 MHz).

- Is highly pattern-dependent for detecting signal to signal shorts.

- Has limited ASIC vendor support.

- Has poor availability of CAD and other development support tools.

- Has difficulty in measuring off-chip current.

6.2.7 Ad Hoc Testing.

Ad hoc testing is a nonbehavioral testability technique. For this testing methodology to succeed, each element of the ASIC must exhibit controllability and observability. It must be possible to control an element's state through the device inputs and observe the operational results at the device outputs. Ad hoc testability techniques achieve these properties by imposing design rules tailored to the particular circuit to be tested. For example, rules may insist that the design provide a means of initializing all storage elements and breaking all feedback loops to permit control of an element's state. Rules may also require that large logic blocks be partitioned into smaller easier-to-test blocks.

Additional ASIC device I/O terminals are required to accommodate the addition of test and control points. The terminals provide tester access to internal circuit nodes for functional control and observation of intermediate results.

Test generation may be easier when special rules, such as logical partitioning are imposed; but, how these rules are implemented and applied are different in each case. These differences result in a custom design and test engineering effort for each ASIC developed. Such efforts are in direct opposition to the reasons for choosing an ASIC solution (Levitt 1992).

Ad hoc testing techniques can be implemented successfully when the gate counts are in the low thousands. Considering the current complexity of ASIC devices, this technique no longer appears to be a viable testing approach.

The advantages of ad hoc testing are as follows:

- Good random logic test capability.
- Low silicon overhead (less than 5 percent).
- Minimal device performance impact.
- Good for highly structured designs.

The disadvantages of ad hoc testing are as follows:

- Requires manual test pattern generation.
- Increases device I/O terminals.
- Requires fault simulation to determine fault coverage.
- May not reduce overall test development time.
- Has no availability of CAD or development support tools.
- Is not supported by ASIC vendors.
- Has a nontransportable test structure and design effort.
- Is suitable for use with only devices with under 10,000 gates.

6.2.8 Behavioral Testing.

Behavioral testing tests the ASIC device in the same way as the device is expected to be operated in system use. All specified functional operations, the sequence of operations, and the associated test data vectors are generated by the ATE. The ATE receives ASIC output data, compares the output data to expected data, and checks the results for errors. The device is exercised by the tester using the same pattern set developed during the design process for functional verification through simulation. The simulator pattern sets are reformatted and translated to operate in the manufacturing tester. Certain tester operations, for example those that model tester behavior, are generally included with the simulation pattern set.

The ATE is generally designed to test the ASIC at the specified system operating speed. Testing is usually performed under the worst case conditions of temperature and supply voltage to weed-out marginal devices. In system operation, the ASIC could be exposed to many operating conditions and sequence of operations not specified or expected. To test the ASIC for all possible conditions would be impractical or impossible. A 100 percent operational test is obtainable with behavioral testing. A fault coverage of 100 percent is not possible to achieve with behavioral testing only. The level of attainable fault coverage from behavioral testing has been reported to be 60 to 80 percent. The percentage of fault coverage decreases as ASIC complexity increases (Strickland 1995).

There are failure modes associated with ASIC devices that can be detected using nonbehavioral testing methods only. Also, there are failure modes that can be detected using behavioral at-device-speed testing only. Most behavioral tests are run at-system-speed which generally is slower than at-device-speed and therefore will not detect these failure modes.

Most ASIC test methods will not detect a fault that causes an increase in the time taken for a specific sequence of tests to propagate from the inputs to the outputs of the device under test (Mascitelli 1994). Generally, these faults can be observed only during at-speed testing. Some examples of device defects that require at-speed testing to be detected are as follows:

- Internal parallel gate defects.
- Resistive vias in signal lines.
- Resistive shorts from the gate to drain.
- Crosstalk between circuits.

Parallel gate configurations are used to increase the drive capability of a circuit when the load demand is greater than the drive availability from one circuit; for example, to support a large signal fan-out requirement. Internal parallel gate defects can reduce the circuit drive capability and cause speed failures because of increased rise times, as illustrated in figure 63. This fault could be caused by a design defect; but more likely, the cause is from a photo masking problem where a contamination particle blocked out a circuit contact on the photo mask. Low speed or dc functional testing would not detect this problem, because the presence of a logic level is checked long after the transition period of the affected output waveform.

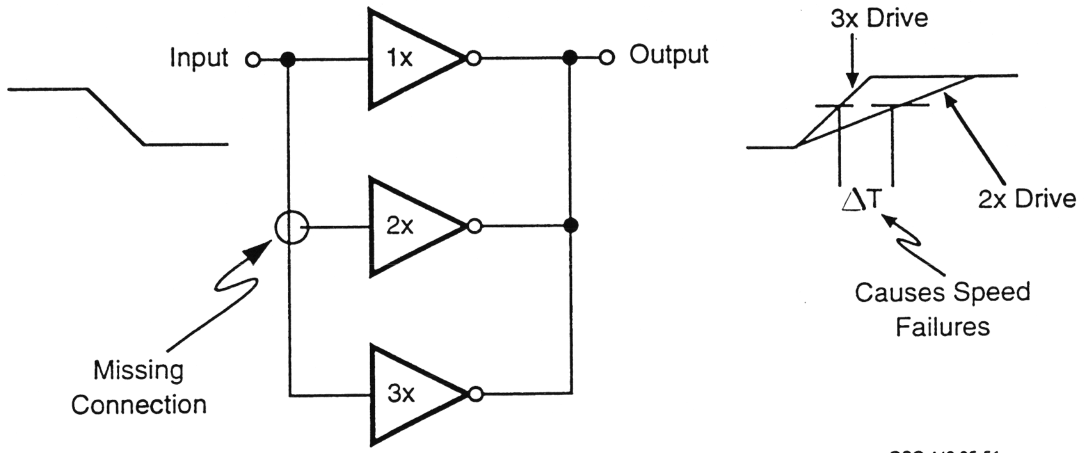


FIGURE 63. MISSING CONTACT FAULT
(Mascitelli 1994)

Vias are used to route signals between metal layers in an IC. Resistive via problems are prevalent in multilayer metal parts. They are caused by oxide remaining in the via in the signal line connection between two metal layers. The via resistance depends on the amount of the oxide remaining and can vary from hundreds of ohms to hundreds of kilo-ohms. Figure 64 shows a two-input NAND gate with a resistive via in series with a signal line. This resistance, coupled with the distributed circuit capacity, forms an RC time constant which will introduce a delay time in the signal path. This signal delay time can vary from a few to many nano-seconds, depending upon the via resistance value.

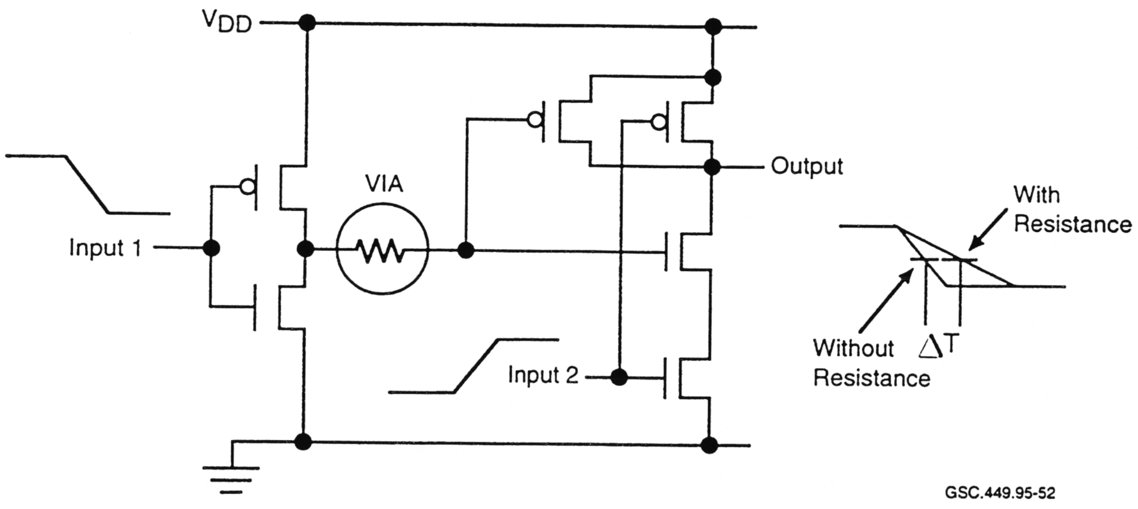


FIGURE 64. RESISTIVE VIA FAULT
(Mascitelli 1994)

Gate oxide is an insulating material and forms the barrier between the source and drain nodes of CMOS transistors. Resistive shorts from a CMOS gate to drain will reduce circuit drive capability and increase circuit switching transition time.

Figure 65 shows a two-input NAND gate with this problem. If the device is not properly protected during handling, static electricity can stress or destroy the device. This is a common problem that can rupture a gate oxide layer and create this type of fault.

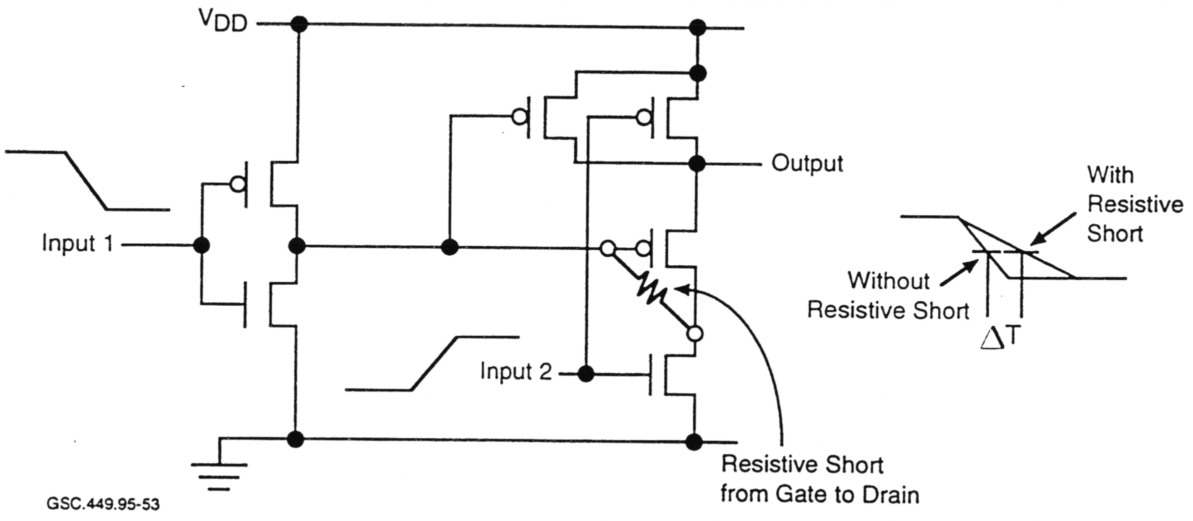


FIGURE 65. GATE TO DRAIN SHORT
(Mascitelli 1994)

Crosstalk between circuits is a problem created by the physical property of the ASIC device layout and not a fault introduced during the manufacturing process cycle.

Figure 66 depicts the coupling between two circuits created by the proximity of two parallel metal lines. These lines may be vertical or horizontal and be located on two layers or on the same layer. Depending on the voltage swing, capacity, frequency, and physical property of the nodes, the coupling may or may not cause a problem. Crosstalk exists in all semiconductor devices; the issue is whether it will cause speed-related problems that produce, for example, false clocking of gates or circuit oscillations.

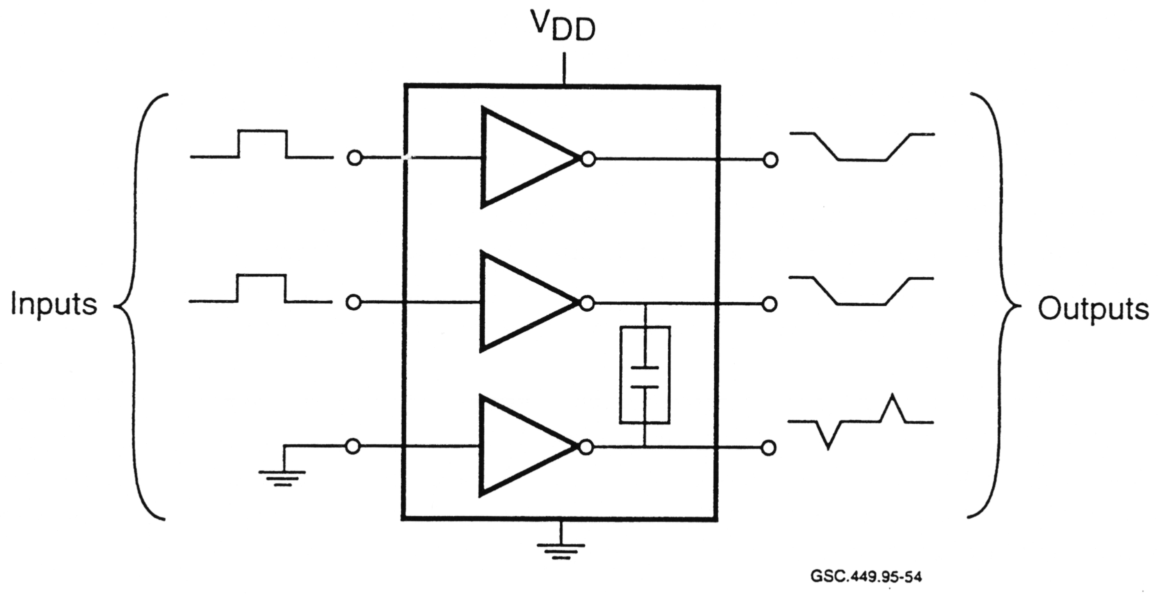


FIGURE 66. COUPLING BETWEEN CIRCUITS
(Mascitelli 1994)

7. APPLICATION SPECIFIC INTEGRATED CIRCUIT VERIFICATION AND VALIDATION.

7.1 THE NEED FOR VERIFICATION AND VALIDATION.

According to Zaidi (1995) "the most difficult task in system-on-chip design is verification of system functionality." There are no standard methods for verification and validation (V&V) of ASICs. Designs using different technologies, architectures, and design techniques require that V&V be tailored to each specific application. A problem with V&V is that it can be haphazard and random if not planned, executed, and monitored thoroughly from the start of the design cycle.

Rapidly evolving technology requires that verification methods also change. Simulation is one of the main verification methods for ASICs. A major problem that verification engineers face is that simulation speed can be very slow. Special tools are sometimes designed in order to speed up specific simulation tasks.

With any complex verification task, it is hoped that all the design errors can be found, but this is not always the case, nor does it mean that passing verification testing results in an error-free design. Verification testing needs to account for the ASIC architecture. Random test vectors are applied but are limited in their ability to detect problems. Controllability and observability of internal logic becomes less visible as more functions are integrated into ASICs. Special verification tests suited to the unique characteristics of each ASIC design are necessary.

Each step in the design, test, and fabrication cycle is equally important. A weak link in any one area is an invitation for disaster. This was demonstrated by the recently discovered Pentium IC flaw. For the Pentium IC, a different technique for floating point division was employed than was used on the 80486 CPU. The technique, called "radix 4 SRT," computed the quotient twice as fast as the conventional shift-and-subtract method, since it generated two bits of quotient for every clock cycle. A plot of partial remainders versus divisors is used to access the next quotient digit by means of a lookup table.

The lookup table was generated manually during the development process, externally to the Pentium IC. In order for the table to be loaded into the hardware, a script is used. A problem in the script caused the inadvertent omission of several table entries. Consequently, any access of the lookup table at these table offsets generated the floating point division error.

While ASICs with design errors that make it into commercial products are not uncommon, this one in particular was the "most widely publicized bug in computer history" (Geppert 1995). Considering the financial consequences for Intel (nearly one-half of a billion dollars), this bug will be remembered for a long time. Are bugs such as this simply a way of life for designs that are complex and difficult to verify? According to Levy (1995):

> ...when you're designing a device with millions of transistors, the chance of reaching absolute perfection is minimal... The question is: When and how will a bug surface?

The Pentium problem demonstrates that each part of the design and development process is important to the integrity of the final product. Not only is it imperative to verify that a design is correct, but also that tools are used correctly and that there are no interface problems between tools or database problems. Editors, compilers, HDLs, synthesizers, placement and routing tools, file conversion tools, and download tools need to be able to pass database information accurately and need to have compatible data formats.

There are a number of simulation approaches used for verification. Traditional simulation is event-driven and uses interpretation of the HDL, and not compilation, for model execution. Every signal value change is scheduled by the simulator and simulation may be paused between any two events. This approach allows a great deal of flexibility for design debugging. However, extra overhead associated with event handling is incurred as a result of this flexibility. Complexity has increased functionality and the number of events which must be processed, resulting in increasingly higher run-times required for simulation.

Cycle-based simulators execute faster since they ignore timing. Blocks of logic, instead of individual events, are scheduled when a signal changes value. On each clock cycle, only those blocks with signal changes are evaluated.

A compiled-code simulator converts the HDL code into an executable language, such as C. This gives the compiled-code simulator a significant speed advantage. As more timing information is added to the simulation, performance degrades significantly.

These verification techniques all rely upon the designers to provide adequate vectors for running the simulations. Since designs are becoming increasingly complex, more time is consumed by designing the test vector set, and running the additional vectors. Available time does not allow complete coverage of a design using test vectors (O'Neill 1995).

A hardware emulator compliments simulation. Multiple large PLDs can be loaded with the design data for a large ASIC. The PLDs are run at hardware speeds and allow the system or diagnostic software to be exercised instead of test vectors. While emulators have fast response times, they do not model the timing accurately. Also, reconfiguring the PLDs for a new design revision can be more time consuming than running a software simulation.

Essentially, the simulations all apply vectors to the input of the model. Events are propagated through the model and output results are recorded. The model can then be executed at a different level, and the results compared, revealing whether or not the two models are equivalent.

"Formal" verification is a verification technique much different from simulation. It is starting to gain recognition among ASIC designers and is being incorporated into certain tools. A tool that has incorporated formal verification builds internal mathematical models of the logic contained in the HDL and netlist design descriptions. Special algorithms are then executed to test if the two designs are equivalent.

7.2 SIMULATION.

Essentially, a simulation executes a model of a real device. Models have been generated and simulations run on device designs from the specification level down to the gate level. Simulation is relied upon to flag errors and verify correct operation at all design levels. Simulation allows rapid design iterations so that changes can be tested easily. Due to the cost involved in rework, detection of errors before the silicon is cast is a high priority for any development effort. Accurate simulations are relied upon to meet these demands.

Although simulation allows rapid design iterations, it is fundamentally a verification technique, not a design technique. While simulators are an integral part of the design environment, they cannot assist designers in the transition from concept to schematic, generate timing or interface specifications, or demonstrate how to fix a timing problem.

Designs are represented in a number of ways when simulating high-level designs. Large designs are generally partitioned and different portions can be represented as follows (Synopsys 1994):

- Black box—No functionality is specified for the black box.
- Bus functional—Bus functional represents functional or statistical behavior. Timing and interrupt handling may be excluded from this representation.
- Behavior/RTL—Behavioral representation accurately describes how a module works and includes clock cycle and other functional data.
- Gate-level netlist—A gate-level netlist includes the full logic implementation with all timing information.
- Hardware modeler—A hardware modeler represents the hardware and allows it to interface to the system hardware. As more details are modeled, the verification time becomes longer.
- Hardware emulation—Hardware emulation, not strictly a simulation technique, allows the RTL description, which was derived from synthesis of the VHDL code, to be downloaded into an emulator. This, in essence, creates a hardware prototype that can verify design functionality at close to actual hardware speeds. An advantage of hardware emulation is that scenarios too complex for execution on a software simulator can be performed. Hardware emulation is further discussed in section 7.4.

7.2.2 Simulation and Validation.

A general definition for validation is given in the glossary of the Digital Systems Validation Handbook-Volume II. It is defined as "the process of evaluating whether or not items, processes, services, or documents accomplish their intended purpose in their operating environment."

For ICs, validation involves establishing proof that the design is what the design team intended to capture. It demonstrates that there are no missing, incorrect, or undesired functions. Validation can be done at a number of different levels. At the HDL level, a design function can be validated by simulation. A gate-level design is necessary to validate speed and area design criteria.

For HDL modules, typically only the functional validation is performed. Simulation is used almost exclusively for validation of today's designs. The input for this simulation can originate at a number of different sources, including VHDL and text files. Later, when verification of the post-synthesis functional design is performed, validation patterns are again applied. This can be a problem, since different tools may have different input formats and requirements. Inexact mapping can also be a source of error introduction.

High-level simulation typically consists of iterations of design and debug cycles. Tools need to be concise in reporting errors. Problems should be identified as clearly as possible. When designing with HDL, a source-level debugger should be used, as is done for many high-level languages. Also, a view of the gate-level design is helpful when correcting errors. This aids the designer in understanding what hardware is produced as a result of HDL coding and changes to that coding.

As with other high-level development environments, the simulation environment should be integrated with other related tools, such as the gate-level design and the source-level debugger. Tools should allow rapid movement between these environments so that the effects of changes can be determined rapidly. Error messages, environmental data such as the directory tree, simulation results, and other such messages should be displayed appropriately.

When the synthesis is performed, validation is necessary at the gate level. A large number of input vectors is needed to obtain reasonable levels of coverage. One of the important considerations is simulation speed. Faster simulation with test vectors allows more of the design to be validated. Methods used to speed simulation include hardware acceleration and the use of multiple CPUs.

A technique called "multi-level simulation" can also be used to speed up the validation process. With multi-level simulation, one portion of the design can be examined at the gate level, while the rest of the design remains at the HDL level. Individual portions of the design can thus be validated. A partitioning of the validation process in this manner can make a complex design much easier to understand and validate.

7.3 ADDITIONAL CONSIDERATIONS FOR SIMULATION.

Simulation tools for HDLs are either event-driven or cycle-based. For each simulation cycle, the event-driven simulator has three or four phases. Also, each clock cycle requires multiple simulation cycles where logic states are evaluated.

In event-driven simulation, a function's inputs may change, requiring re-evaluation. Other functions may not be evaluated if their inputs did not change. It is also necessary to maintain an ordering algorithm for event-driven simulators. This is to ensure that events and evaluations occur in the correct order.

Since most ASIC designs are synchronous, the circuits change value only on the active edge of a master clock. The simulator partitions the ASIC internals into two parts: flip-flops and combinatorial elements. The simulator propagates all combinatorial functions within one clock cycle. A cycle-based simulator requires only one cycle per clock, and during that cycle there is only one evaluation phase. The cycle-based simulator does not need to consider time elements, as does the event-driven simulator. Hence, cycle-based simulation is much faster than event-driven simulation.

Cycle-based simulators have traditionally used proprietary languages. This means additional learning time in the design cycle. For early cycle-based simulators, another problem was that designs could not be moved between different simulators. When EDA simulation languages became standardized, subsets of event-driven languages were used in the cycle-based simulators. However, today, test benches and system-level models often use constructs that are not supported by most cycle-based simulators. Therefore, the models may require a rewrite for use with a cycle-based simulation.

Some other factors need to be considered when working with simulation tools. These are

- Simulation accuracy—For design, as well as for verification, the accuracy of the simulation model is important. Submicron designs are at the greatest risk of not having accurate simulation parameters. This is due primarily to submicron phenomena that profoundly influence signals. Other factors that can diminish the simulation accuracy include:

  - incomplete modeling of the external stimulus,
  - inaccuracy in the model of any circuit component,
  - failure to detect problems in the simulation results, and
  - inaccuracy in the test bench.

- Simulation test vectors—How are test vectors entered for simulation? Manual and graphical inputs are time-consuming and error prone. When inputs are simple or repetitive, these methods are adequate. Complex and nonrecurring patterns are more difficult.

- Simulation complexity—Design complexity is a key factor when considering simulation. For some designs, under normal conditions, the proportion of a design that can be verified through simulation, due to time or system limitations, may be unacceptable. The size of the database may reduce the efficiency of the simulation platform. An alternative may be to use a simulation platform that can perform hardware acceleration. A decision to use hardware acceleration may be based on the following:

  - How many simulation cycles are required?
  - Will the database size degrade performance and impact the overall schedule?
  - How much time has been allocated for simulation?
  - What are the potential risks of performing a smaller number of simulations?

- Simulation results display—When performing verification, results are displayed on the CRT in a number of formats. These display formats include:

  - Waveform—Gives a graphical representation of selected signals.

  - Interactive—Allows simultaneous viewing and updating of schematic and waveform displays.

  - Tabular—Formats listings based on clock cycles.

  - Custom—Meets user-specific display requirements.

7.4 APPLICATION SPECIFIC INTEGRATED CIRCUIT PROTOTYPING AND VALIDATION.

It was only a few years ago when hardware prototyping was deemed an essential part of a development task. However, a rapidly emerging technology brought about a number of changes that would impact design prototyping negatively. The major changes include:

- ICs were becoming more complex and the silicon was not as easily testable.
- Due to the increased physical complexity, IC tester interfaces were much more difficult to design.
- Many new package types were introduced in order to handle the numerous I/O pins counts.
- TTM pressures from competitors meant that testing needed to be largely automated.

The cost of prototyping was getting expensive, and the associated delays were unacceptable. Due to these changes, the amount of prototyping performed decreased dramatically. In order to test designs, simulation tools were coming into widespread use. The designer was able to check the design without having to produce the silicon IC first. Changes were easily compiled and checked since there was no hardware fabrication involved.

There are many advantages of simulation runs for checking designs before committing to actual hardware. A software simulation is better for checking implementations that are broken into submodules. These submodules, having limited I/O, are easier to understand and test.

There are drawbacks as well. Some types of architectures and designs that contain algorithms are difficult to test with software-based simulators. For instance, without a prototype to test an algorithm thoroughly, there will be little data available to verify its correct operation. Response to unpredictable and asynchronous stimuli cannot be checked on simulators. The more complex a design, the more likely it is to contain errors. Hence, the more likely it is to benefit from prototype testing.

Factors that help designers decide whether to simulate or prototype a design include:

- Difficulty of testing the design using software-based tools.
- Time available for testing.
- Amount of testing that is required.
- Risk tolerance for errors that simulation may not reveal.

Due to the drawbacks, designers have started to use prototyping once again for some applications. However the hardware platform used for prototyping has also changed significantly. Instead of fabricating a prototype IC, the design is emulated using programmable logic, such as CPLDs or FPGAs. These devices have densities greater than 50,000 gates and are reprogrammable. Some have software-reconfigurable logic, allowing logic changes at CPU speeds.

Designers should seriously consider prototyping when there is a long build time for the silicon, or there are dire financial or safety issues. Whether to prototype or not should not be based on the probability of an error in the logic, but rather its consequences if undetected. In assessing risk, following are pertinent questions:

- Will correct design operation be ensured if it is implemented without logical errors?
- Are there submodules in the design that have not been used previously?
- As data move through the submodules, do they follow easily testable bounds?
- Are data understood and testable between and at the boundary conditions?
- Which will cost less, a redesign or a prototype?
- Does the schedule allow time for a redesign?

Since simulation does not verify correct design and operation adequately in certain cases, prototyping can be an effective addition to the verification suite. For safety-critical systems, prototyping should be a requirement.

7.5 FORMAL METHODS FOR HARDWARE DESIGN.

Formal verification uses an analytical approach for proof of correctness and does not rely on the use of test vectors, as with simulation. The method is inherently thorough, as opposed to test vectors, which in practice do not provide full coverage. Many tools now incorporate formal methods so that this verification technique does not need to be performed as a separate, manually intensive task.

Formal verification takes only minutes to execute on current tools and checking small revisions is not a problem for this technique, as it is for simulation. Formal verification results highlight faulty logic, where simulation does not. Formal verification also facilitates verification of modules. A drawback of formal verification is that it does not verify timing. It is best to use formal verification along with a static timing-analysis tool.

7.6 SOFTWARE REUSE.

Software reuse is an issue for designers. Once code is developed, it may be possible to use large portions of the code in a different application. This is easily done for software, since code is portable. Hardware, on the other hand, has not shared this capability. However, with the use of HDLs, logic designs, or portions thereof, can be reused. In fact, when ASIC designs contain tens of thousands of lines of code, there is significant motivation for developers to reuse as large a portion of that code as possible. Development cost can be reduced and the TTM shortened for products where HDL reuse is implemented.

There are several ways that hardware can be reused. First, if the HDL code is well-written, structured, partitioned, and well-documented, blocks of code may be suitable for reuse. Functions such as counters, comparators, and ALUs are constructed by designers and are used commonly in design.

Second, libraries containing numerous logic functions are available with synthesis tools. These libraries are designed to increase productivity by providing designers with standard functions.

Common available functions include:

- AND gate,
- AND-OR gate,
- D flip-flop,
- D flip-flops with scan logic,
- J-K flip-flop,
- Latch, and
- Multiplexer.

Numerous variations on these basic functions are available from tool vendors.

Third, logic cores are also available. These can include error correction, data compression, digital signal processing, CPU, memory, and other highly integrated logic functions. One manufacturer offers ASIC core logic containing (Mayer 1995):

- Intel 80C31 microprocessor,
- MIL-STD-1553 data bus controller,
- MIL-STD-1750 microprocessor,
- A reduced instruction set computer,
- 54XX series logic elements, and
- RAM.

More options in core logic are available from vendors as ASIC technology progresses. Designing with core logic is an easy way to produce a complete system on a single IC. As geometries continue to reduce, more core logic will be included in ASICs. Design time is lessened significantly if core logic can provide the required functionality.

For software, certification credit has been a consideration for code that had been certificated previously as part of an avionic system. A question arises as to whether digital hardware designs will be treated in the same manner as software. For instance, since current complex ASIC designs are essentially software projects, should portions of the HDL code that have been certificated previously as part of a larger system receive credit for that certification?

7.7 REGULATIONS AND GUIDELINES.

On the regulatory and design guidance side, the Federal Aviation Regulations (FARs), RTCA/DO-160C - Environmental Conditions and Test Procedures for Airborne Equipment, SAE ARP1834 - Fault/Failure Analysis for Digital Systems, and Advisory Circulars (ACs), address issues from specific perspectives. RTCA/DO-160C addresses issues relating to the environment in which airborne equipment must operate. It examines numerous environmental influences, including:

- temperature,
- altitude,
- humidity,

- shock and vibration,
- power and power line issues,
- radio frequency susceptibility and emission, and
- lightning.

While the RTCA/DO-160C testing guideline is necessary, it is not the only guideline that can be applied to LRUs. Failure analysis on avionic equipment can be accomplished using procedures in ARP1834. The need for failure analysis techniques is pointed out in AC 25.1309-1A and FAR Parts 23, 25, 27, and 29, section 1309, and is implied by FAR Part 33, section 75. ARP1834 has been adopted as an informal guideline for meeting these requirements.

ARP1834's analyses are specifically meant to identify digital equipment hardware faults. ARP1834 is not an exhaustive or universally accepted method for applying fault/failure analysis. It is used merely to present cost-effective, industry-acceptable means for identifying failure modes and failure effects. Manufacturers who wish to use ARP1834 as a certification guideline should discuss their reasoning with the regulatory agency early in the process. This is because variations of approaches presented in ARP1834 will need to be employed under different circumstances. For systems that are flight-critical or flight-essential in nature, one approach might be to develop design techniques for a fault tolerant system.

However, testing to RTCA/DO-160C and ARP1834 is not, by itself, sufficient to ensure that failures will be extremely improbable, as required for flight-critical systems. While avionics manufacturers and airframers do use additional tests and design assurance methods, currently there are no guidelines that address the life-cycle considerations for complex ICs. Thus, the certification process lacks uniformity. What is deemed sufficient by one ACO may not meet the demands of another ACO. Additionally, manufacturers need to know what to expect during the certification process.

While it is obviously in the best interest of the avionics supplier to provide reliable equipment, the methods manufacturers use vary. Reliability and safety issues, as applied to the development processes of complex ICs designed for safety-critical applications, are in dire need of being addressed. In an effort to standardize the system certification process for hardware, and address current hardware issues, RTCA has formed Special Committee 180 (SC-180).

7.8 RTCA SPECIAL COMMITTEE 180.

In an effort to address hardware issues at a systems level, RTCA is coordinating a working group of government and industry experts. This working group, SC-180, is tasked with the formation of a document that will serve as a guideline for industry during the development of airborne hardware systems. Specifically, it will focus on design assurance methods for airborne hardware. This guideline will be modeled after RTCA/DO-178B. As such, it will not provide specific certification requirements for hardware systems but will provide general guidelines for meeting those requirements.

A listing of the terms of reference generated by SC-180 are found in appendix A. The terms of reference provide guidelines for SC-180 in the formulation of the new document. The focus and bounds of the document are stated, and relationships with other related standards organizations are stipulated.

Comprehensive design guidance for airborne hardware systems is necessary. It will benefit both industry and FAA in detailing standard procedures for meeting certification criteria. CEs should realize, however, that following the mechanics of design guidance does not in any manner guarantee defect-free hardware or systems. It simply means that the outlined process was followed. There are a number of areas such as design techniques, technology-specific requirements, and testing techniques that will not be addressed in the design guidance, yet they are essential ingredients in the development of error-free and defect-free hardware. It is the developer's task to use design tools that consider all aspects of a particular technology and apply the most comprehensive test techniques to ensure design integrity.

8. CONCLUSION.

Current fly-by-wire technology necessitates greater concern for safety issues relating to digital flight control and avionic systems. Fly-by-wire aircraft are using ASICs to implement flight-essential and flight-critical functions. Threats to device reliability exist and need to be addressed in complex IC designs. These complex ICs include devices such as FPGAs and ASICs. While classically the ASIC was not user-programmable, in essence, it now is. An engineer with the right tools on a PC can do the entire design, simulation, test generation, and then send it to the silicon foundry for fabrication and packaging.

There are many advantages to be gained by using complex user-programmable devices, such as ASICs, but a closer examination is warranted when flight-critical or -essential systems with user-programmed complex ICs are presented for certification. It is not a simple matter to demonstrate that these devices are designed correctly or tested adequately.

8.1 A NEW LOOK AT AN OLD TECHNOLOGY.

The use of digital technology in aircraft is nothing new. Implementations of early digital logic ICs were relatively easy to analyze and their failure modes were well understood. While the use of ASICs in aircraft is seen as a benefit for avionics manufacturers and airframers, its implementation raises concerns about the safety of systems in which they are used. Part of the problem is due to the sheer complexity of current ASIC devices. Failure analysis guidelines that were developed for digital systems, such as SAE's ARP1834, cannot be applied in a meaningful way to the complex ICs that are being designed today. Additionally, new failure modes, that were not a problem for older digital technology, are now prevalent and can compromise the safety of systems using these complex devices.

Commercial fly-by-wire aircraft are now being produced that have differing design philosophies from earlier aircraft. While digital avionics and flight controls have existed for a number of years, there were always backup systems that relied on a different technology, in case of a failure of the digital system. These were hybrid aircraft using a combination of control hydraulics with interfaces to digital systems. Today's fly-by-wire aircraft, as typified by the Boeing 777 and Airbus A320, use data buses to send actuation commands to the various control surfaces based on messages generated from the avionic systems. On these aircraft, the hydraulic link no longer exists. It is expected that if there is a failure on a primary system, another system with duplicate capabilities and connectivity will be available to take control. Back-up systems are simply duplicates of the primary systems. Redundancy may sometimes be implemented using dissimilar hardware and software, and integrity is enhanced by voting systems and other techniques, but the technology remains the same.

Due to increased IC densities, ASICs can now be programmed to take on tasks that were formerly performed in software. For instance, communication protocols are implemented in a chip. HDLC and ARINC 629 are complex transactions that are described in a written specification and implemented in silicon. It is no trivial matter to verify (1) that the protocol is completely and correctly specified in written form, (2) that it was implemented completely and correctly in silicon, and (3) that the silicon is not defective. At current ASIC complexity levels, it is dangerous to assume that upset avionics are due solely to software bugs.

It is vital to ensure that these digital avionic systems are designed correctly and tested fully. This is no trivial matter. Error-free parts can no more be guaranteed than one can promise error-free software. In fact, complex ASIC design is described by Corcoran (1995) as a "software project being performed by hardware engineers," since most ASIC parts are now designed using high-level languages that describe digital logic behavior. The following sections will highlight and discuss some of the problem areas that may be encountered when working with complex ASICs.

8.2 CERTIFICATION.

It is anticipated that almost all hardware in the future will be composed of ASICs, FPGAs, and other user-programmable and special function ICs. Sound engineering practice necessitates the development and use of a process to guide design, development, and test toward meeting specific design requirements and safety goals.

Avoiding scrutiny for a flight-critical system component by seeking a hardware solution that is less encumbered by regulations is a way for developers to cut costs, but the overall safety impact should be examined carefully. Fundamental verification issues can be bypassed with a silicon-based implementation.

Certification of systems containing ASICs is a potential problem for developers and certification engineers. Currently, there are no techniques and methods of design, documentation, testing, and verification identified or recognized by the FAA for today's complex hardware designs. Existing guidance does not address current practice or technology. For instance, ASIC designs are implemented using a full suite of computer-based tools that are not regulated by any guidelines, while tools used in software development are. (RTCA/DO-178B requires tool qualification for software tools in safety-critical applications.)

Work is underway through RTCA's SC-180 to develop hardware design assurance guidance for digital systems, although it is not anticipated that the resulting guidance will address all the issues. Questions concerning hardware/software integration, hardware reuse, hardware/software codevelopment, and other issues are likely to remain. While process guidance is necessary, certification specialists should keep in mind that new testing and design verification methods are still emerging in this rapidly evolving technology. New methods that enhance device controllability and observability, along with more rigorous verification methods should be encouraged.

8.3 DESIGN.

With ASIC technology, design and coding is not an isolated function but relates to each step in the development and test phases. A change necessitated by simulation, synthesis, test, or layout necessitates a change in the HDL code. Complex ASIC design is not really hardware design as it once existed. ASICs are now designed using software. VHDL is a structured language, and coding proficiency is not achieved easily. A complex ASIC can require a hundred thousand lines of high-level code. In the future, as complexity increases, this number will grow significantly.

As a rule-of-thumb, designs with more than 10,000 gates are not done by schematic entry. Tools become a necessity above this level. Due to constant rapid developments in digital technology, however, tools are still playing catch-up to the technology, leaving designers with little support in some areas of design and test. Tool-induced design errors occur and can be difficult to detect. The fidelity and completeness of simulation tools is essential to design integrity, yet modeling deep-submicron phenomena is an area that lags behind current technology. Being able to predict accurately signal propagation delays can mean the difference between a properly operating device, and one that is marginal or failing.

8.4 TESTING AND VERIFICATION.

Thorough device testing is critical to the quality of an ASIC and, for safety-critical systems, to the safety of the user. The goal for test is to ensure that no faults exist which can cause the device to malfunction. Any limitation in test capability can mask problems and allow defective devices to be installed in a system which eventually may fail in operation. If the failure is in a safety-critical system, the effects must be noted and corrected by the system or system operator, or the results could be disastrous to both life and property.

Automotive and medical electronics companies demand 100 percent fault coverage for ASICs used in vehicle control and human life support systems. Demanding a fault coverage of 100 percent for ASICs used in safety-critical avionics is only the beginning in achieving safety. A good quality management program that will follow the ASIC development from design through installation and maintenance is essential. This program includes personnel, development and test equipment, supporting systems, procedures, documentation, environment, and the ASIC devices.

All ASIC faults must be detectable in manufacturing test. Many circuits can be tested by performing at-system-speed test diagnostics that simulate the way the device is used in the system. This behavioral testing method generally can provide 60 to 80 percent fault coverage. The remaining circuits are either impractical or impossible to test using this method and are tested using nonbehavioral DFT methods which are designed into the ASIC.

Several well-supported DFT methodologies are currently available. Each has its own set of advantages and disadvantages to consider. Which is best to use depends on the requirements and design specifications of the particular ASIC being developed. If no single testing method will provide 100 percent fault coverage, multiple techniques must be used. One method that always should be used in testing complex ASICs is I_{DDQ} testing. Approximately 10 percent of all ASIC faults are detectable only with I_{DDQ} testing.

Test synthesis tools are providing the capability for anyone knowing the application to design an ASIC using functional descriptive techniques. Meaningful verification may be difficult to demonstrate since knowledge of all development steps and their associated products is necessary. In order to detect and identify ASIC faults, IC engineers and process specialists must be part of the development team. Certain faults, resulting from the existence of physical phenomena in the ASIC, are undetectable and unknown to the designers during synthesis. Generally, these faults can be detected using at-device-speed behavioral test methods.

Even ICs, whose failure can cause substantial financial penalties to the manufacturer, such as the Intel Pentium, are not immune from process errors that can cause defective hardware. There was no design error in Intel's Pentium microprocessor, but a step in the development process had allowed bad data to slip into the design. Even seemingly insignificant process steps that are well understood and rarely have been the source of problems can be a source of failure. This error cost Intel close to $500,000,000.00 (Geppert 1995). It serves to show that errors sometimes do find their way into hardware, even though this hardware underwent some of the most rigorous verification processes in industry.

Unlike software, ICs are influenced by changes in temperature, voltage, noise, and other environmental variables. Combinations of fluctuations in these variables can induce failure. Temperature changes can produce timing skew; voltage changes produce temperature changes and also change noise immunity characteristics. Thorough device testing within the manufacturer's specified operating environment is impractical.

There are failure modes associated with ASICs that are not readily identifiable. They can be the result of design errors or subtle phenomena that are not flagged by the tool suite. These phenomena include clock skew, ground bounce, and crosstalk. ASICs can exhibit data-sensitive behavior due to the cumulative effects of internal currents resulting from peculiar data patterns. These errors may not be found during testing due to the impracticality of complete pattern testing for a complex ASIC. If these ASICs are part of an avionic system, they may increase the number of LRUs removed for servicing and lead to more "no fault found" results at the test bench.

It can be difficult to detect faulty operation of an ASIC. Complex ASICs require that test circuitry be designed into the ASIC. Designs for fly-by-wire aircraft require fault tolerant architectures, not simply redundancy. There are no FAA guidelines that would suggest to airframers how this is done or what to require of their avionics suppliers.

Due to the current necessity to create ASICs and FPGAs using software (i.e., at a much higher level of design abstraction than in the past), systems engineers and even programmers are designing ASICs. Meaningful verification may be difficult to demonstrate since knowledge of all development steps and their associated products is necessary.

8.5 COMPLEXITY ISSUES.

According to Keller (1992),

> Some electronics systems themselves have attained a complexity level such that traditional lab and flight testing methods cannot realistically be extensive enough to show compliance to existing requirements.

This is certainly true for ASICs, which are at the core of modern avionics. ASICs can contain embedded microprocessor cores with user-supplied software. Complex ASICs can replace complete systems containing PROM (for software memory), CPUs, digital signal processors, RAM, and other random logic elements.

Typically ASICs are programmed by the end-user or avionics supplier. Complex ASIC designs can require teams of 50 to 100 engineers. ASICs are a technology essentially unregulated by the FAA and not understood by certification engineers. The level of on-chip circuit elements that can be squeezed into an ASIC is so high that more of the software portions of avionic system designs are being placed into ASICs. ASICs are used extensively on the Boeing 777 in flight-critical systems and, along with other user-programmable logic and special function ICs, will be used almost exclusively in future fly-by-wire aircraft.

Predicting reliability for ASICs involves more than simply predicting the probability of a hardware failure. Classic reliability figures assume that the device has no design errors and that the silicon has no defects. For complex ASICs, these assumptions may not be true. ASIC designers cannot guarantee error-free designs, and ASIC manufacturers cannot guarantee error-free silicon. ASICs used in flight-critical systems can contain latent faults, having unpredictable effects.

Another issue which complicates reliability prediction is the fact that complex ASICs are now created using high-level software. In order to arrive at a more accurate estimate of ASIC reliability, software reliability issues may also need to be taken into account.

Some may feel that in order to avoid the burgeoning complexity and potential problems associated with deep-submicron design, this technology should be avoided altogether for safety-critical systems and only larger ASIC geometries should be used. This approach may be possible, but only for a little while. As the semiconductor industry continues to invest in new technologies, there comes a point when the older technologies are no longer supported simply due to economic considerations. This obsolescence seems to be occurring at a rapid pace. It is therefore doubtful that avoiding deep-submicron design could be anything more than a short-term avoidance of an inevitable problem.

ICs eventually fail. Therefore system architectures will always include redundancy. Fault tolerant architectures should be in place, not only at the systems level, but also internal to the ASICs. In order to manage design complexity and ensure that each internal functional block can be tested thoroughly, sufficient partitioning of internal logic needs to be included. Architectures that facilitate fault identification such as parity, watchdog timers, variable limit checking, and other such schemes should be in place to flag incorrect device behavior.

9. BIBLIOGRAPHY.

Agrawal, Om P., "Fast, Dense, Predictable and 100% Routable MACH 3 & 4 Family," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Agarwal, Vinod K., "ESTA Takes BIST for a Fast Drive," Electronic Engineering Times, June 1995.

Aitken, Robert C., "An Overview of Test Synthesis Tools," IEEE Design & Test of Computers, Volume 12, No. 2, Summer 1995.

Aitken, Robert C., "Fault Location with Current Monitoring," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Allison, Jeffrey, "Designing ASICs for First-Time Success at Cisco." ASIC & EDA, February 1994.

Altera Data Book, Altera Corporation, San Jose, CA, August 1993.

Altera Data Sheet, "MAX 7000 Programmable Logic Device Family," Version 2, Altera Corporation, San Jose, CA, August 1994.

Altera Data Sheet, "FLEX 8000 Programmable Logic Device Family," Version 4, Altera Corporation, San Jose, CA, August 1994.

Amador, Eric and Ken L. Nelsen, "Source Level VHDL Design Techniques," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Ambler, Tony, et al., "The Economics of Design for Test," EE-Evaluation Engineering, Volume 33, No. 11, Nokomis, FL, 1994.

Anderson, Thomas L., "Adding Partial-Scan to an Existing Design Flow: A Case Study," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Anderson, Thomas L. and Christopher K. Allsup, "Incorporating Partial Scan," Asic & EDA, October 1994.

"Are 5-ns CPLDs the End of 22V10s?" ASIC Design Insert, Computer Design, August 1994.

Arnold, Bill, "Standard Cells Pace CMOS ASIC Growth." ASIC & EDA, March 1994.

ARP1834, Fault/Failure Analysis for Digital Systems and Equipment, Society of Automotive Engineers, Warrendale, PA, August 7, 1986.

Atmel Corporation, CMOS Data Book, San Jose, CA, 1993.

Avra, L. J., et al., "High-Level Synthesis of Testable Designs: An Overview of University Systems," Test Synthesis Seminar - Digest of Papers, IEEE International Test Conference, Altoona, PA, October 1994.

Baker, Scott, "Development of Image Resizing ICs Using a Structured Custom Design Method," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Banerjee, Prithviraj and Jacob A. Abraham, "Characterization and Testing of Physical Failures in MOS Logic Circuits," IEEE Design & Test, August 1984.

Bartlett, Donald, "Consider Testability When Designing Mixed-Signal ASICs," EDN, November 1993.

Baze, M.P., et al., "Single Event Upset Test Structures for Digital CMOS Application Specific Integrated Circuits," IEEE Transactions on Nuclear Science, Volume 40, No. 6, December 1993.

Beal, Bill and Hans Cramer, "Understanding the Electrical Properties of Bond Wires and their Effect on Signal Integrity," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Beechler, Robert K. and Robert Hanz, "The Role of Vendor-Specific Programmable Logic Tools," ECN, Volume 38, No. 10, October 1994.

Bennetts, R. G., "Test Synthesis - Many Things to Many People," IEEE Design & Test of Computers, Volume 12, No. 2, Summer 1995.

Bier, Jeff, "Benchmarking Programmable DSPs: A New Approach," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Blazej, Daniel C., Bruce E. Kurtz, and Bernard P. Gallomp, "Enabling Technologies for Achieving Avionics Modernization," Proceedings of the IEEE 1988 National Aerospace Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Bowen, Jonathan P. and Michael G. Hinchey, "Ten Commandments of Formal Methods," Computer, IEEE computer Society, Volume 28, No. 4, New York, NY, 1995.

Breuer, Melvin A., ed., Design Automation of Digital Systems, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972.

Brittain, Keith, Dave Dalnodar, and Michael R. Sottile, "Unique Applications of Custom MIL-STD-1553 ASIC," IEEE/AIAA 12th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1993.

Broseghini, James, "Circuit Design Techniques for Low Power Circuits," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Buchner, T., B. Hofflinger, and T. Schwederski, "WSP - A Processor for Real-Time Wheel Slip Measurement in Vehicles," 41st IEEE Vehicular Technology Conference, Gateway to the Future Technology in Motion, IEEE, New York, NY, 1991.

Bunnel, James C., "Saving Power in On-Chip Design," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Carmi, Llan, John Thomson, and Carol Bagdis, "Top-Down Design Strategy Includes DFT," Integrated System Design, Los Altos, CA, February 1995.

Carroll, Michael, VHDL-Panacea or Hype?, IEEE Spectrum, June 1993.

Chattopadhya, Sandip and Ken Ward, "Design Reuse: Merging Legacy and Modern Design Methodologies," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Cheng, Kwang-Ting, "Single-Clock Partial Scan," IEEE Design & Test of Computers, Volume 12, No. 2, Summer 1995.

Chenoweth, David and Mark Muegge, "Embedding Virtual Test Points in PCBs," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Chickermane, Vivek, "DFT: Test Synthesis and Beyond," Test Synthesis Seminar - Digest of Papers, IEEE International Test Conference, Altoona, PA, October 1994.

Chou, Tai-Yu, "Signal Integrity Analysis in ASIC Design," ASIC & EDA, May 1994.

Cohen, Moshe, S., "Use of Statecharts in a HW/SW Co-Design Environment," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Conner, Doug, "Submicron Technologies Require Floorplanning," EDN, September 1993.

Corcoran, Tim, "Top Down ASIC HDL Modeling Methodology," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Cornfield, P.S., "ASIC Design for Communications Satellite Payloads," IEEE Colloquium Digest, 1993/41, 1993.

Coston, Terry W., "A Status Report on Analog HDLs," Integrated System Design, Los Altos, CA, November 1994.

Cox, Henry, "Synthesizing Circuits with Implicit Testability Constraints," IEEE Design & Test of Computers, Volume 12, No. 2, Summer 1995.

Curran, Jim, Trends in Advanced Avionics, Iowa State University Press, 1992.

Cypress Semiconductor, Programmable Logic Data Book, Cypress Semiconductor Corporation, San Jose, CA, July 1994.

Daniel, Wayne, "Optimizing Fault Detection for Boundary Scan Testing," Integrated System Design, Volume 7, Issue 75, September 1995.

Design Automation Newsletter, Design Automation Technical Committee, IEEE and IEEE Computer Society, Spring 1995.

Digital Design Verification Symposium, Tektronix, Inc., Ft. Washington, PA, October 1994.

Digital Systems Validation Handbook, Volume II, DOT/FAA/CT-88/10, U.S. Department of Transportation, Federal Aviation Administration, May 1989.

Dislis, C., et al., "The Economics of Chip Level Testing and DFT," Test Synthesis Seminar - Digest of Papers, IEEE International Test Conference, Altoona, PA, October 1994.

Donlin, Mike, "Adopting New DFT Tools Still Calls for Caution," Computer Design, Volume 34, No. 4, April 1995.

Donlin, Mike, "IC Complexity Pushes EDA into the Test Arena," Computer Design, Volume 33, No. 6, May 1994.

Donlin, Mike, "Standards Groups Vow to Solve Tool Interoperability Problems," Computer Design, Volume 34, No. 6, June 1995.

Donnay, S., et al., "A Methodology for Analog Design Automation in Mixed-Signal ASICs," Proceedings of the European Design and Test Conference, IEEE Computer Society Press, Los Alamitos, CA, 1994.

Dudzinski, Rod, "Signal Crafting to Maintain Signal Integrity," EDN, Volume 40, No. 12, Highlands Ranch, CO, June 1995.

Dyson, Amy and Richard Leach, "Power PC™ Architecture Equals Powerful DSP," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

EDA Select Catalog, Mentor Graphics, Wilsonville, Oregon 1994.

Edirisooriya, Geetani, Samantha Edirisooriya, and John P. Robinson[1], "A New Built-In Self-Test Method Based on Prestored Testing," The Eleventh IEEE VLSI Test Symposium, IEEE Computer Society Press, Los Alamitos, CA, April 1993.

Edirisooriya, Geetani, Samantha Edirisooriya, and John P. Robinson[2], "Time and Space Correlated Errors in Signature Analysis," The Eleventh IEEE VLSI Test Symposium, IEEE Computer Society Press, Los Alamitos, CA, April 1993.

Ehlscheid, Steve, "A Practical Method to Increase Test Coverage Using I_{DDQ}," EE Evaluation Engineering, Volume 34, No. 8, August 1995.

Eichenlaub, Steve and Aaik van der Poel, Sr., "Getting the Best from an FPGA via Logic Synthesis," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Ensell, James J., "Rapid Prototyping for Core-Based ASICs Approaches, Trade-Offs, and a DSP-Core Case History," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Evans, David L., "It's Time to Build a VHDL Infrastructure," Military and Aerospace Electronics, August 1994.

Fawcett[1], Bradley K., "Extending FPGA Architectures to Address New Applications," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Fawcett[2], Bradley K., "Synthesis for FPGAs: An Overview," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Fawcett[3], Bradley K., Tools to Speed FPGA Development, IEEE Spectrum, November 1994.

Fawcett[4], Bradley K., "Using Reconfigurable FPGAs in Test Equipment Applications," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Ferguson, Joel F. and Tracey Larrabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS ICs," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Fleming, Peter, "Applications of IEEE Std. 1149.1: An Overview," The Test Access Port and Boundary Scan Architecture, IEEE Computer Society Press, Los Alamitos, CA, 1990.

Fox, Jeffrey R., "A Higher Level of Synthesis," IEEE Spectrum, March 1993.

Freeman, Jeff, et al., "The 68060 Microprocessor Functional Design and Verification Methodology," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Freeman, Jeff, et al., "The 68060 Microprocessor Logic Design and Verification Methodology," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Fritzemeier, Ronald R., et al., "Increased CMOS IC Stuck-At Fault Coverage with Reduced I_{DDQ} Test Sets, Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Gajski, Daniel D. and Loganath Ramachandran, "Introduction to High-Level Synthesis," Design and Test of Computers, IEEE Computer Society Publications, Volume 11, No. 4, Los Alamitos, CA, 1994.

Gajski, Daniel D. et al., High-Level Synthesis, Introduction to Chip and System Design, Kluwer Academic Publishers, 1992.

Gallant, John, "Deep-Submicron Geometries Dictate New Approaches to ASIC Design," EDN, Volume 40, No. 12, Highlands Ranch, CO, June 1995.

Gannot, Gary and Michiel Ligthart, "Logic Synthesis for Programmable Logic Design," Exemplar Logic, Berkley, CA, 1994.

Geppert, Linda, "Biology 101 on the Internet: Dissecting the Pentium Bug," IEEE Spectrum, February 1995.

Gheewala, Tushar, "Structured Test is Free," ASIC & EDA, July 1994.

Gladstone, Bruce E., "Everything But the Chip," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Gloster, Clay S. and Franc Brglez, "Boundary Scan with Built-In Self-Test," The Test Access Port and Boundary Scan Architecture, IEEE Computer Society Press, Los Alamitos, CA, 1990.

Goldberg, Jeffrey, "Achieving 100% Routability with 100% Utilization in a Complex PLD Architecture," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Golnabi, Habib and John D. Provence, "A Technique for Implementing BIST," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Graham, J. Scott, Matthew S. Thompson, and Robert A. Johnson, "A GaAs ASIC Chip Set for NASA's Deep Space Network," Proceedings of Fifth Annual IEEE International ASIC Conference and Exhibit, IEEE Service Center, Piscataway, NJ, 1992.

Gruebel, Robert, "Using DFT in ASICS," EE-Evaluation Engineering, Volume 34, No. 3, March 1995.

Gupta, Rajesh K., Claudionor N. Coelho, Jr., and Giovanni De Micheli, "Program Implementation Schemes for Hardware-Software Systems," Computer, Volume 27, No. 1, January 1994.

Halliday, Andy and Greg Young, "Test Synthesis - System to Foundry," Test Synthesis Seminar - Digest of Papers, IEEE International Test Conference, Altoona, PA, October 1994.

Harris, Ian G. and Alex Orailoglu, "Fine-Grained Concurrency in Test Scheduling for Partial-Intrusion BIST," Proceedings of the European Design and Test Conference, IEEE Computer Society Press, Los Alamitos, CA, 1994.

Harrison, Lee H. and Peter J. Saraceni Jr., "Certification Issues for Complex Digital Hardware," Proceedings of the 13th Digital Avionics Systems Conference, Institute of Electrical and Electronic Engineers, New York, NY, 1994.

Hawkins, Charles E., "I_{DDQ} Design and Test Advantages Propel Industry," IEEE Design & Test of Computers, Volume 12, No. 2, Summer 1995.

Hawkins, Charles E., et al., "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Haydamack, William, "The Reshaping of EDA," ASIC & EDA, August 1994.

Hecker, Ramone, "FPGA ASIC Emulation of 25K Gate Design," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Hemmady, Shankar, "VLSI Test Automation: Fact or Fiction?" ASIC & EDA, September 1994.

Hirzer, J., "Testing Mixed Analog/Digital ICs," The Test Access Port and Boundary-Scan Architecture, IEEE Computer Society Press, Los Alamitos, CA, 1990.

Hofer, David, "Automated Critical-Path Testing in ASIC and IC Designs," EE-Evaluation Engineering, Volume 34, No. 9, September 1995.

Hold, Betina, P.C.P. Bhatt, and V.K. Agarwal, "Rapid Prototyping and Synthesis of a Self-testing ABS Controller using CAD Tools," Proceedings of the IEEE Workshop on Real-Time Applications, IEEE Computer Society Press, Los Alamitos, CA, 1994.

Hronik, Stanley, "Industry Acceptance of Boundary Scan," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Huijbregts, Ed. P., Jos T.J. van Eijndhoven, and Jochen A.G. Jess, "On Design Rule Correct Maze Routing," Proceedings of the European Design and Test Conference, IEEE Computer Society Press, Los Alamitos, CA, 1994.

Huling, George, Introduction to use of Formal Methods in Software and Hardware, Conference Record, Wescon/94 idea/Microelectronics, IEEE Operations Center, Piscataway, NJ, 1994.

IEEE Standard 1149.1, Test Access Port and Boundary-Scan Architecture, Institute of Electrical and Electronic Engineers, May 21, 1990.

Jain, Prem P., "IC Design for an Image Processing Algorithm," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Jacob, Gerald, "One Step Closer to Achieving Structured Analog DFT," EE-Evaluation Engineering, Volume 33, No. 12, December 1994.

Jani, Dhanendra and John M. Acken, "Test Synthesis for Microprocessors," Test Synthesis Seminar - Digest of Papers, IEEE International Test Conference, Altoona, PA, October 1994.

Janowitz, Joan, "Handbook-Volume III Digital Systems Validation Book Plan," DOT/FAA/CT-93/16, July 1993.

Jessen, Eric and Bob Hutchins, "Converting Multiple FPGAs into a Single ASIC," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Josephson, Doug, et al., "Microprocessor I_{DDQ} Testing: A Case Study," IEEE Design & Test of Computers, Volume 12, No. 2, Summer 1995.

Kapusta, Richard, "Options Dot the Programmable Logic Landscape," EDN, Volume 40, No. 14, July 1995.

Karpel, Ron, "Ultraspec: A Cycle-Based Simulator Engine for VHDL," ECN, Volume 39, No. 8, August 1995.

Kassab, Mark, Janusz Rajski, and Jerry Tyszer, "Accumulator-Based Compaction for Built-In Self Test of Data-Path Architectures," First Asian Test Symposium Proceedings, IEEE Computer Society Press, Alamitos, CA, November 1992.

Keating, Mike, and Dennis Meyer, "A New Approach to Dynamic I_{DDQ} Testing," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Keller, Forrest L., "Basic Electronic Systems Certification," Proceedings of the IEEE/AIAA 11th Digital Avionics Systems Conference, October 1992.

Keller, John, "Airframers Grapple with SEU Problem in Avionics," Military and Aerospace Electronics, July 1993.

Khalilollahi, Yousef, "Switching Elements, the Key to FPGA Architecture," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Khatakhota, Mehdy, "A Practical Solution for High Speed On-chip Clock Distribution," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Kirk, Bob and Peter Fletcher, "ASIC Prototyping with FPGA's," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Kirk, Bob, "FPGA Migration to ASIC's," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Knack, Kella, "Testward, Ho," ASIC & EDA, November 1993.

Koenemann, Bernd, et al., "Test Synthesis and Beyond," Test Synthesis Seminar - Digest of Papers, IEEE International Test Conference, Altoona, PA, October 1994.

Koga, R., et al., "The Impact of ASIC Devices on the SEU Vulnerability of Space-Borne Computers," IEEE Transactions on Nuclear Science, Volume 39, No. 6, December 1992.

Krambeck, Robert H., "Efficient Use of .5µm Technology with ORCA," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Kumar, Dilip, Young Lee, and Samuel Lee, "3D Graphic System—ASIC Implementation Using High Level Design Methodologies," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Lapsley, Phil, "Low Power Programmable DSP Chips: Features and System Design Strategies, On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Le, Chinh H., "A Case for On-Chip Design Re-use," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

LeBlanc, Johnny J., "LOCST: A Built-in Self-Test Technique," IEEE Design & Test, November 1984.

Lee, Kuen-Jong and Melvin A. Breuer, "Constraints for Using I_{DDQ} Testing to Detect CMOS Bridging Faults," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Levitt, Marc E., "ASIC Testing Upgraded," IEEE Spectrum, May 1992.

Levy, Markus, "Intel Cures the Pentium Blues," EDN, Volume 40, No. 2, January 1995.

Ligthart, Michiel, "Logic Synthesis for Programmable Logic Design," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Lipman, Jim, "Submicron EDA Tools Help Tackle Tough Designs," EDN, Volume 40, No. 12, Highlands Ranch, CO, June 1995.

Loo, Thomas, "Testability Considerations in High-Performance Avionics Processors," IEEE Design & Test, April 1992.

Lorusso, Stephen M. and Michael T. Fertsch, "Integrating CrossCheck Technology Into the Raytheon Test Environment," Test: Faster, Better, Sooner, IEEE International Test Conference, Altoona, PA, 1991.

Makhijani, Haresh and Stephen Meier, "A High Level Design Solution for FPGAs," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Malaiya, Yashwant K., et al., "A detailed Examination of Bridging Faults," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Malaiya, Yashwant K., et al., "Modeling and Testing for Timing Faults in Synchronous Sequential Circuits," IEEE Design & Test, November 1984.

Maly, Wojchieck and Marek Patyra, "Built-in Current Testing," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Mano, M. Morris, Computer System Architecture, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

Martuscello, Anthony R., "Designing DSP Systems," ASIC & EDA, March 1994.

Mascitelli, Alida, "At-Speed ASIC Testing, What Types of Defects Will It Detect?" EE-Evaluation Engineering, Volume 33, No. 11, November 1994.

Maston, Gregory A., "Production Constraints on Test Synthesis," Test Synthesis Seminar, IEEE International Test Conference, Altoona, PA, October 1994.

Maunder, Collin M. and Rodham E. Tulloss, "Boundary Scan - A Review," The Test Access Port and Boundary-Scan Architecture, IEEE Computer Society Press, Los Alamitos, CA, 1990.

Mayer, John H., "Low Power, High Performance Drive Military ASIC Development," Military and Aerospace Electronics, Volume 6, No. 7, July 1995.

McClure, Michael, "Top-Down Timing Design Techniques," EDN, Volume 40, No. 16A, Highlands Ranch, CO, August 1995.

McEuen, Stephen D., "I_{DDQ} Benefits," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

McHugh, Patrick F. and Lee Whetsel, "Adding Parity and Interrupts to IEEE Std. 1149.1," The Test Access Port and Boundary-Scan Architecture, IEEE Computer Society Press, Los Alamitos, CA, 1990.

Meek, Ivan C., "A Fast Graphic Design Entry and Placement Technique for Field Programmable Gate Arrays," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Merkelo, Henri and Haw-Jyh Liaw, "Design Issues for On-Chip Signal Integrity at High Speed and High Complexity," Design SuperCon 95 Conference Paper, University of Illinois, Urbana, IL, 1995.

Mermet, Jean, VHDL for Simulation, Synthesis, and Formal Proofs of Hardware, Kluwer Academic Publishers, 1992.

Messenger, Charles G., "Qualified Manufacturer's List (QML), A New Approach for Qualifying ASICs," Proceedings of the Third Annual IEEE ASIC Seminar and Exhibit, Kenneth W. Hsu and Mark E. Schrader, eds., IEEE, New York, NY, 1990.

Micallef-Trigona, R., "Datapath Intensive ASIC Design - Synthesis from VHDL," IEE Colloquium Digest, 1993/076, 1993.

Miczo, Ph.D., Alexander and Robert D. Duell, "PDV - Path Directed Design Verification," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Miller, Warren, "Real World Applications for Field Programmable Gate Array Devices - An Overview," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Mitchell, Richard, "In-System Programming, the Future for High-Density Devices," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Monaco, James E. and John R. Kasha, "An Automatic Simulation Environment for PowerPC™ Design Verification," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Morley, Sean P. and Ralph A. Marlett, "Selectable Length Partial Scan: A Method to Reduce Vector Length," Test: Faster, Better, Sooner, IEEE International Test Conference, Altoona, PA, 1991.

Mortazavi, Mohammad and Nikolaos Bourbakis, "A Synthesis and Floorplanning Methodology for Multiple-Chip Module (MCM) Design," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Motohara, Akira and Hideo Fujiwara, "Design for Testability for Complete Test Coverage," IEEE Design & Test, November 1984.

Munson, John C. and Ruth H. Ravenel, "Designing Reliable Software," Fourth International Symposium on Software Reliability Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1993.

Nigh, Phil and Wojciech Maly, "Test Generation for Current Testing," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Nuñez, Ramon, "Design Verification: Finding the Needle in the Haystack," Events, IKOS Systems, Cupertino, CA, Spring 1995.

O'Donnell, Gary, "Using 1149.1 for Multi-Drop and Hierarchical System Testing," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

O'Neill, Brian, "Formal Verification Rapidly Checks ASIC Design Revisions," Computer Design, Volume 34, No. 8, Tulsa, OK, August 1995.

Oakland, Steven F., et al., "An ASIC Foundry View of Design and Test," Test Synthesis Seminar - Digest of Papers, IEEE International Test Conference, Altoona, PA, October 1994.

Olen, Mark and David Hoffer, "Automating Boundary Scan Design," Wescon/94 Conference Record, IEEE Operations Center, Piscataway, NJ, September 1994.

Olsen, Glenn, "System Design Considerations for FPGA Synthesis," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Ouellett, Timothy R., "Development and Using PeaRLs (Pre-Routed Logic)," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

PAL Device Data Book and Design Guide, Advanced Micro Devices, Sunnyvale, CA, 1995.

Parker, Alice C. "Automated Synthesis of Digital Systems," IEEE Design and Test of Computers, IEEE Computer Society, Volume 1, No. 4, Los Alamitos, CA, 1984.

Petropoulos, Leo, "Arithmetic Functions in Programmable Logic," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Petropoulos, Leo, "Prototyping Beats Simulation for Complex Real-Time Designs," EDN, Volume 40, No. 12, Highlands Ranch, CO, June 1995.

Petropoulos, Leo and Glenn Quiro, "Boundary Scan Testing and In-System Programming Aid in the Use of CPLDs," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Pivin, David P., "ASICs According to Darwin," Electronic Component News, Volume 39, No. 3, March 1995.

Poplin, Dwight, "ASIC Tests Require Vectors to Test Possible Stuck Conditions," EDN, Volume 40, No. 2, January 1995.

Porter, M., et al., "A Testable EnDec/Data Synchronizer/Frequency Synthesizer Device," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Prioste, Jerry E., "Design Considerations and Solutions for High Performance CMOS System Interfaces," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Programmable Logic Data Book, Xilinx, Inc., San Jose, CA, 1994.

Rajan, Sundar, "Practical State Machine Design Using VHDL," Integrated System Design, Los Altos, CA, February 1995.

Rearick, Jeff, "Synthesis for Testability Versus Design for Testability in a Full Custom Design Environment," Test Synthesis Seminar, IEEE International Test Conference, Altoona, PA, October 1994.

Reizenman, Michael J., "Technology 1994, Test and Measurement," IEEE Spectrum, January 1994.

Ridgeway, David J. "Designing PCI Local Bus Interfaces with Programmable Logic," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Rose, Susan, "EDN News Breaks - EPROMs Evolve to Meet Specific Application Needs," EDN, October 1992.

Roth, Charles, Tom McDonald, and Rob Mains, "Timing Analysis and Optimization of the PowerPC™ 604 Microprocessor," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Roy, Rabindra K., et al., "Test Synthesis Using High-Level Design Information," Test Synthesis Seminar, IEEE International Test Conference, Altoona, PA, October 1994.

RTCA/DO-160C, "Environmental Conditions and Test Procedures for Airborne Equipment," Radio Technical Commission for Aeronautics, Washington, DC, December 1989.

RTCA Paper No. 535-93/SC180-11, "777 Application Specific Integrated Circuits Certification Guideline," Washington, DC, December 1993.

RTCA Paper No. 185-94/SC-180-23, "Terms of Reference," Joint RTCA Special Committee 180 and EUROCAE Working Group 46, Washington, DC, April 1994.

Rudnick, Elizabeth M., et al., "Sequential Circuit Testability Enhancement Using a Nonscan Approach," IEEE Transactions on VLSI Systems, Volume 3, No. 2, IEEE Computer Society Press, Los Alamitos, CA, June 1995.

Runyon, Stan, "I_{DDQ}: Does it work?" Electronic Engineering Times, August 1995.

Rushby, John, "Formal Methods and their Role in Digital Systems Validation for Airborne Systems," Digital Systems Validation Handbook, Volume III, Chapter 1, DOT/FAA/AR-95/125-III,1, Draft, U.S. Department of Transportation, Federal Aviation Administration.

Rushby, John, Formal Methods and Digital Systems Validation for Airborne Systems, CR-4551, National Aeronautics and Space Administration, Hampton, VA, December 1993.

Ryherd, Eric, "Prototyping ASIC Core Functions," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Sadowski, Greg, "An HDL to FPGA Path for VIDEO Applications," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Saunders, Larry and Yatin Trivedi, "A Picture is Worth a Thousand Gates," ASIC and EDA, Los Altos, CA, November 1994.

Schediwy, Bic, "High Density Reprogrammable Logic Devices Take on Gate-Array Class Application," Electronic Component News, November 1993.

Schneider, K., Th. Kropf, and R. Kumar, "Control Path Oriented Verification of Sequential Generic Circuits with Control and Data Path," Proceedings of the European Design and Test Conference, IEEE Computer Society Press, Los Alamitos, CA, 1994.

Schulz, Steven E., "Ready for Prime Time?" Integrated System Design, Los Altos, CA, April 1995.

Sedmak, Richard and Colin Munder, "Benefits and Penalties of Boundary Scan," The Test Access Port and Boundary-Scan Architecture, IEEE Computer Society Press, Los Alamitos, CA, 1990.

Sebaa, Lahouari, et al., "Self-Test Video Controllers," Wescon/94 Conference Record, IEEE Operations Center, Piscataway, NJ, September 1994.

Shao, Gary, Steve Thielker, and William Hanna, "Generic ASIC Design: A Sensible Methodology for Rapid Insertion of VHSIC/VLSIC Technology," Proceedings of the IEEE 1988 National Aerospace Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Shaw, John L., Hans K. Herzog, and Kenji Okubo, "Digital Autonomous Terminal Access Communication (DATAC)," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, October 1986.

Sherman, Jeffery, "To Use BSDL Successfully, Validate Device Descriptions Carefully," EDN, Volume 40, No. 12, June 1995.

Simtimes, "FPGAs Offer New Design Challenges," Synopsys Logic Modeling Group, Beaverton, OR, January 1995.

Sivaraman, Mukund and Andrzej J. Strojwas, "Towards Incorporating Device Parameter Variations in Timing Analysis," Proceedings of the European Design and Test Conference, IEEE Computer Society Press, Los Alamitos, CA, 1994.

Smith, Donald, "Delving Into Deep Submicron," Integrated System Design, Los Altos, CA, February 1995.

Steinberg, Carol J. and Ian M. Wilson, "Simulation Programs Iron Out Transmission-line Effects," Microsim Corporation, Irvine, CA, 1994.

Strickland, T., "Design-for-Test Methodologies and Tools for ASIC Design," Electronic Products, Volume 38, No. 1, June 1995.

Stroele, Albrecht P., "Reducing BIST Hardware by Test Schedule Optimization," First Asian Test Symposium Proceedings, IEEE Computer Society Press, Alamitos, CA, November 1992.

Stroele, Albrecht P., "Signature Analysis for Sequential Circuits with Reset," Proceedings of the European Design and Test Conference, IEEE Computer Society Press, Los Alamitos, CA, 1994.

Stroud, Charles E., "Built-In Self-Test for High-Speed Data-Path Circuitry," Test: Faster, Better, Sooner, IEEE International Test Conference, Altoona, PA, 1991.

Sugita, Gary, "Enhancements to the World's Fastest CPLD Family Give Designers More Flexibility," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Sylvestri, John and Peter Ouimet, "I_{DDQ} Test As a VLSI Failure Analysis Tool," EE-Evaluation Engineering, Volume 34, No. 5, May 1995.

Synopsys, "Making the Transition to High-Level Design," Synopsys Inc., Mountain View, CA, June 1994.

Takla, Mourad B., Donald W. Bouldin, and Daniel B. Koch, "Early Exploration of the Multi-dimensional VLSI Design Space," Proceedings of the Seventh International Conference on VLSI Design, IEEE Computer Society Press, Los Alamitos, CA, 1994.

Taub, Edward S., "ASIC Verification Across Multiple Development Environments," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Tegethoff, Mick M.V., "IEEE Std. 1149.1: Where are we? Where From Here?" IEEE Design & Test of Computers, Volume 12, No. 2, Summer 1995.

Thompson, D.W., J.T. Harrington, and K.A. Parker, "A Macrocell Approach to Disk Drive Electronics Integration," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Tiwary, Gyanendra, "Reducing Power Consumption in ASICs," Computer Design, Volume 34, No. 3, March 1995.

Turino, Joe, "Lifetime Cost Implementations of Chip-Level DFT," Test Synthesis Seminar, IEEE International Test Conference, Altoona, PA, October 1994.

Tustin, Wayne, "Recipe for Reliability: Shake and Bake," IEEE Spectrum, December 1986.

Vaidya, Chaitanya, "Design and Application of High-Speed Clock Generators," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Velazco, Raoul, "Test and Diagnosis of Programmable Integrated Circuits - PhD Thesis, N92-11253, NASA Center for Aerospace Information, Linthicum Heights, MD, November 1990.

Vereen, Lindsey, "Where are the Standard Parts of Yesterday?" ASIC & EDA, September 1994.

"VHDL Now U.S. Government Standard," Military and Aerospace Electronics, Volume 4, No. 2, February 1993.

VM1553 Data Manual, Government Products Division, VLSI Technology, Inc., 1990.

VLSI Technology, Inc., Short Form Catalog, March 1993.

Waller[1], Larry, "A Tale of Two ASICs," Integrated System Design, Los Altos, CA, February 1995.

Waller[2], Larry, "And in this Corner," Integrated System Design, Los Altos, CA, January 1995.

Warmke, Doug, "Building Up Chips Using VHDL and Synthesis," Integrated System Design, Los Altos, CA, January 1995.

Warp2 VHDL Development System User's Manual, Cypress Semiconductor, San Jose, CA, 1994.

Wasson, Stephen L., "Floorplanning Xilinx FPGA Designs For High Performance," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Watson, Lynn R., "Raising Abstraction of ASIC Design with 'Pins-Out'," On-Chip System Design Conference, Day 1, Hewlett-Packard Company, 1995.

Weiss, Ray and Julie Anne Schofield, "EDN's 19th Annual μP/μC Chip Directory," EDN, November 1992.

Widman Associates "Introduction to Designing for Synthesis," Seminar, Wescon 94 Idea/ Microelectronics Conference, Anaheim Convention Center, Anaheim, CA, September 1994.

Wilson, Ron and David Lammers, "IBM, NEC Aim Huge Gate Arrays at High End of Market," Electronic Engineering Times, Issue 833, January 1995.

Winkel, David and Franklin Prosser, The Art of Digital Design, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980.

Winters, Mike, "Using IEEE-1149.1 For In-Circuit Emulation," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Won, Martin S. and Craig Lytle, "Continuous Interconnect in the FLEX 8000 Architecture," Wescon 94 Conference Record, IEEE Operations Center, Piscataway, NJ, 1994.

Wright, E. Frederick, Using an Embedded RISC to Test ASICs," On-Chip System Design Conference, Day 3, Hewlett Packard, 1995.

Wright Laboratory, Wright Laboratory Success Stories, 1993.

Wu, Yuejian and Ivanov Andre, "Reducing Hardware with Fuzzy Multiple Signature Analysis," IEEE Design & Test of Computers, Volume 12, No. 1, Spring 1995.

Yovits, Marshall C. ed., Advances in Computers, Volume 37, Academic Press, San Diego, CA, 1993.

Zaidi, S. Jauher A., "System-On-Chip Design and Verification in Mass Storage," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

Zeidman, Robert, "The Basics of ASICs Tutorial," Anaheim Convention Center, Anaheim, CA, 1994.

Zorian, Yervant, "BIST for Embedded Memories," EE-Evaluation Engineering, Volume 34, No. 9, September 1995.

10. GLOSSARY.

ALIASING. A condition where good signatures generated by the test data compression process are indistinguishable from the bad signatures. Aliasing is generally caused by an error in the data compression algorithm.

APPLICATION SPECIFIC INTEGRATED CIRCUIT. A semi-custom chip used in a specific application. ASICs are designed by integrating standard cells and arrays from a library.

ASTABLE. Describes a circuit or system that has no stable state. Such a system will oscillate. Astable circuits can be used to generate timing and synchronizing clock signals.

ASYNCHRONOUS. 1. Describes a sequential logic system wherein operations are not synchronized to a common clock. 2. Describes signals whose behavior and timing are unrelated to a particular clock. Signals are based on known but random events whose timing cannot be precisely predicted.

AUTOMATIC TEST PATTERN GENERATOR. A software program that, given a circuit description and a list of faults, automatically generates the test vectors necessary to detect the faults specified for that circuit.

BEHAVIORAL TESTING. A method of testing that exercises the device in the same way it will be used in the target system.

BISTABLE. Describes a system or circuit that has two stable states. Any other state is unstable and will eventually change to one of the stable states.

BOUNDARY REGISTER. A register added to cells to combine signals at the boundary between core modules or core modules and the I/O terminals. The boundary register collects data from and presents data to modules on the boundary during test.

BUILT-IN SELF-TEST. A design method that allows a device to test itself by adding logic for test signal generation and analysis of test results.

BUILT-IN TEST EQUIPMENT. The actual hardware or software which is built into a device to provide built-in self-test capability.

CHIP. A single piece of semiconductor material which contains integrated circuitry. A chip is also referred to as a substrate or die.

CLOCK SKEW. A variation in the arrival time of the active clock edge between two or more clocked flip-flops. Clock skew can result in incorrect logic values from the affected circuits.

COMPLEMENTARY METAL OXIDE SEMICONDUCTOR. An integrated circuit used for memory and logic cells. It uses negative-well MOS (NMOS) and positive-well MOS (PMOS) transistors configured in a complementary pair fashion that results in low power operation.

COMPUTER-AIDED DESIGN. Specialized software used for architectural, mechanical, or electrical design.

COMPUTER-AIDED ENGINEERING. Specialized software used for analyzing designs created and entered into a computer. Engineering analysis includes for example, electronic circuit analysis.

COMPUTER-AIDED MANUFACTURING. Specialized software used for computer-controlled manufacturing operations. CAD information is input to this system.

CONTROLLABILITY. The ability to set a node inside a device to a desired value via an input pin. For a digital circuit, the value would correspond to a logic 1 or 0.

CROSSTALK. The unwanted capacitive or inductive coupling of signals between adjacent conductors or circuits.

DATA COMPACTION. A technique for reducing the space required for data storage. Also known as data squishing.

DATA PATH. A portion of a design that typically comprises arithmetic and word-wide logical operations.

DESIGN FOR TESTABILITY. The design of a device to enhance its controllability and observability and thereby ease test generation.

DIE. Same as a chip, particularly before being placed in an IC package.

ELECTRONIC DESIGN AUTOMATION. Software and hardware tools used to ascertain the viability of an electronic design. These tools perform simulation, synthesis, verification analysis, and testing of the design.

ELECTRONIC SYSTEMS DESIGN AUTOMATION. Graphical front end software and hardware tools that allows the use of pictures rather than words to describe and analyze a design. These tools provide a higher degree of abstraction over traditional schematic capture or waveform display programs.

ELECTROSTATIC DISCHARGE. The natural physical event of the transferring of electrical charges. If uncontrolled, ESD can destroy semiconductor devices which have inadequate packaging and handling protection.

FAULT COVERAGE. The number of detectable device faults divided by the total number of possible device faults expressed as a percentage.

FAULT DETECTION. The ability to determine if a fault is present.

FAULT DIAGNOSIS. The ability to locate a fault and determine its cause.

FAULT SIMULATION. The process of using software to simulate the operation of a circuit to which faults have been intentionally added so that the ability of a set of test vectors to find the faults can be determined.

FIELD PROGRAMMABLE GATE ARRAY. A logic device that is programmable and has a high density of gates.

FINITE STATE MACHINE. A machine which can be in one of a finite number of states. Often used for a logic circuit which sequences through various states. Such a circuit is referred to as sequential.

FLIP-FLOP. 1. A bistable digital circuit. 2. An electronic circuit having two stable states and two inputs corresponding to the two states. The circuit remains in one state until caused to change to the other by the application of the corresponding input signal. The two states are referred to as the set and reset state, or a logic 1 and logic 0 state.

FLOATING GATE. A gate on a metal oxide semiconductor transistor that is not connected to anything.

FORMAL VERIFICATION. Verifying electronic circuit functionality using mathematical proofs.

FRAMEWORK. A software infrastructure that provides a common environment for the communications and integration of tools in a process.

GROUND BOUNCE. A ringing (damped oscillation) on an output signal when one or more outputs on the same device switch from logic 1 to logic 0.

HARDWARE DESCRIPTION LANGUAGE. A specialized programming language that describes the physical design, electronic behavior, logic structure, and system annotation information for circuits. The language allows design description at a high level of abstraction while supporting a logical synthesis path to gate level implementation.

INTEGRATED CIRCUIT. An electronic device which has many transistors and other semiconductor components integrated into one piece of silicon.

LEVEL-SENSITIVE SCAN DESIGN. Discipline for structural design for testability. Level-sensitive refers to constraints on circuit excitation, logic depth, and handling of clocked circuitry. Scan refers to the ability to shift any state into or out of the network.

LOGIC CELL. The generic term for a basic building block of a general-purpose logic device.

LOGIC SIMULATION. A means whereby a logic design can be evaluated on a computer before actually being built. The computer simulates the behavior of the components to predict the behavior of the overall circuit.

LOGIC SYNTHESIZER. A compiler module that uses several algorithms to minimize gate count, remove redundant logic, and utilize the device architecture as efficiently as possible.

MASK PROCESS. A series of steps used to cover or coat a semiconductor surface to conceal specific areas for selective depositing or etching.

METALIZATION. The process of connecting the various elements of an integrated circuit by placing a layer of metal over the entire wafer and then selectively etching away unwanted metal. A photolithographic mask defines the pattern of connections.

NETLIST. A data file generated by the design synthesis process that describes integrated circuit functionality as a list of circuit elements and their network of interconnections. The netlist is used by the foundry in the placement and routing process.

NON-BEHAVIORAL TESTING. A method of testing an integrated circuit to see that each logic element and interconnection performs its defined function, regardless of how the device will be used.

OBSERVABILITY. The ability to determine the value at a circuit node inside a device using its external pins. For a digital circuit the value would correspond to a logic 1 or 0.

OXIDE ISOLATION. An integrated circuit technique that uses silicon oxide to isolate transistors. The result is higher speed and density.

PROCESS YIELD. The number of devices produced that test good divided by the total number of devices tested, expressed as a percentage.

PROGRAMMABLE LOGIC DEVICE. A logic device programmed at the customer site. It contains various configurations of gates and flip-flops.

QUIESCENT CURRENT. The power supply current drawn by a circuit after the inputs have changed state and the circuit is in the steady-state non-switching condition. The IEEE symbol for quiescent current flow in a CMOS device is I_{DDQ}.

QUIESCENT CURRENT TESTING. A test methodology used in the testing of CMOS devices to improve the detection of defects and failure mechanisms. The test is performed by measuring the device supply current flow I_{DD} in the quiescent logic state. This test method is referred to as I_{DDQ} testing.

REGISTER TRANSFER LEVEL. Descriptions that are characterized as system definitions in terms of registers, switches (multiplexers), and operations. They are also known as data-flow descriptions.

SCAN CHAIN. A design for testability technique where the storage elements are connected together in a serial shift register chain. This facilitates testing because it simplifies the on and off loading of data into all the sequential elements in a design.

SCAN DESIGN. A design method in which special circuits are used to convert a sequential circuit to a combinational one to ease test generation.

SCHEMATIC ENTRY. The process of describing a circuit by entering a detailed logic diagram into a computer system.

SEQUENTIAL LOGIC. A logic circuit whose operation depends on both present input signals and previous operations or states. The logic requires memory elements for remembering past states.

STATE ENCODING. Finding an optimal binary encoding of an abstract state-machine description.

STRUCTURAL TESTING. A form of testing used to verify the operation of each cell and respective interconnections internal to an IC. Structural testing provides a degree of fault coverage unattainable from functional testing only.

STUCK-AT FAULT. A physical defect that causes a circuit node in a device to remain at a fixed level. For a logic circuit, the level would correspond to a logic 1 or 0.

SYNCHRONOUS. 1. A sequential logic system wherein all operations are synchronized to a common system clock. 2. Signals whose behavior and timing are synchronized to a clock.

SYNTHESIS. Translation and optimization of a hardware description language specification into a gate level implementation.

TEST ACCESS PORT. A device interface controller used to control the access and function of built-in test hardware. The interface is defined by IEEE standard 1149.1-1990.

TEST STRUCTURE. The nature and organization of the test program, including: the origin, form, and type of test data; the destination of the results; and the procedures used to control test operations and process data.

TEST VECTOR. A pattern of bits applied to a circuit in test to detect a fault. A test vector is also referred to as a test data pattern.

VERTICAL TEST INTEGRATION. A testing structure that provides for testing capability of the system, the sub-systems, the PC boards in each sub-system, and the major devices on each PC board. Testability of each device is integrated into the system test structure. An integrated hierarchical test architecture referred to as Electronic System Test Automation, and The Fourth Generation Test Methodology.

VIA. An interconnection between insulated metalization layers of an integrated circuit used to provide a conductive path between layers. Vias provide the same function as plated-through holes provide on printed circuit boards.

WAFER. A round slice of pure silicon which is used in the fabrication of integrated circuits. Many circuits can be built on one wafer. The present standard wafer diameter is 30 centimeters.

INDEX

APPENDIX A—RTCA SPECIAL COMMITTEE 180 TERMS OF REFERENCE

1. The Special Committee should consider the industry experience gained by manufacturers of electronic hardware as well as the results of recent research.

2. Consider existing standards for possible adoption or reference.

3. Establish design assurance levels for hardware and the associated nature and degree of analysis, review, documentation, test, and other design assurance activities. The guidance criteria should be objective in form to support demonstration of compliance with airworthiness requirements.

4. Consider the criteria for qualification of tools to be used for certification credit, for example, tools for hardware design, configuration management, and verification.

5. Establish the assurance criteria to be used for airborne electronic hardware and off-the-shelf hardware not developed under these guidelines.

6. Consider configuration control guidelines, design process assurance guidelines and their compatibility with existing airworthiness requirements.

7. The guidelines should consider and be appropriate for current, new, and evolving technologies.

8. Coordinate with Systems Integration Requirements Task Group (SAE SIRT) to ensure the definition of the information flow between the system and hardware design processes, and maintenance of these interfaces over the development life-cycle, is in accordance with the Aerospace Recommended Practice 4754 document.

9. The guidelines should address the design change process throughout the life cycle of the hardware.

10. Establish design assurance guidance to verify that the hardware design will perform its intended function with a level of confidence that allows compliance with airworthiness requirements.

11. Establish design assurance guidance to verify that existing hardware will operate properly in an airborne application which may be different than the original application.

12. Establish configuration control criteria for optional features of hardware which may be activated or used by different applications, and establish verification criteria for these features or ensure that they do not interfere with the intended function.

13. Recognize the international implications of the document and establish a close relationship with any EUROCAE working group established to address this subject.

14. The guidelines should consider and be appropriate for current, new, and evolving processes such as automatic test pattern generation, logic synthesis, and other automated design and verification processes.

15. The guidelines should consider the aspects of produceability, testability, and maintainability in the design of airborne electronic hardware, as applicable to airworthiness requirements.