Emulation Systems Based on Reconfigurable Hardware Devices

by

Richard Wieler

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree of

Master of Science

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

ISBN 0-612-13563-2

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

ELECTRONICS AND ELECTRICAL ENGINEERING — SUBJECT TERM

0544 — SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES

American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences

Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

EMULATION SYSTEMS BASED ON

RECONFIGURABLE HARDWARE DEVICES

BY

RICHARD WIELER

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Manitoba to reproduce this thesis by photocopying or other means, in whole or in part, at the request of other institutions or individuals for the purpose of scholarly research.

ABSTRACT

Reconfigurable hardware, has traditionally been used to replace a group of static logic on a circuit. The reprogrammable nature of these devices, specifically Field Programmable Gate Arrays (FPGAs) has not been fully exploited. The research carried on in support of this thesis is a part of a greater scope to find unique applications for reprogrammable hardware.

Fault emulation is a novel application of reprogrammable hardware. Two implementations of a fault emulator are presented here. The first, attempts to use the dynamically reconfigurable architecture of certain FPGAs to perform fault insertion using partial reconfiguration. The second implementation uses scalable high capacity FPGAs. This methodology switches in fault models, that have been built into the emulated design.

A secondary objective of this thesis is to examine some of the characteristics of various Built-In Self-Test (BIST) techniques. The two techniques examined are based on Linear Feedback Shift Registers (LFSRs) and Circular Self-Test Path (CSTP) methodologies.

ACKNOWLEDGEMENTS

I would like to express my thanks and appreciation to my advisor Bob McLeod. His openness to always make time to answer my many questions, and his support and advice throughout my time at University is greatly appreciated. Most of all I have to thank him for allowing me to play hockey with him on Thursday afternoons. Thanks also must go to my peers, whose advice and suggestions along the way helped make everything work and make sense. Thanks go especially to Zaifu Zhang, Dean McNeill, Roger Ng, Mike Gamble and Dave Blight. Also a special thanks to Glenda for taking the time to proof read this entire manuscript.

Acknowledgement must at this point also be expressed to Georg Simon Ohm for stating that:

$$V=IR$$

which essentially sums up all of electrical engineering.

A great deal of the credit for the fact that I have make it this far in life goes to my family for their support and encouragement. Thanks especially to Mom and Dad, I could never have made it through without your love and encouragement and support. Now that I'm moving away I'm going to miss you all.

TABLE OF CONTENTS

CHAPTER 3

STATIC RECONFIGURABLE EMULATION

CHAPTER 4

CONCLUSIONS AND FUTURE WORK

APPENDIX A

TESTING OVERVIEW

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Purpose

In the past the traditional approach to logic design has followed one of two routes. The first was to use discrete logic chips, which required considerable area overhead, but was reasonably affordable. The second alternative was to build an Application Specific Integrated Circuit (ASIC) which greatly reduced the area, but was substantially more costly for low volume production. As technology further developed, Programmable Logic Devices (PLDs) became a cost effective alternative for some ASIC and discrete logic designs. The initial PLDs were based on arrays of logic that could be interconnected. The interconnect technology was limited to two families: fused based and transistor based. The transistor based programming technology includes: Erasable Programmable Read Only Memory transistors (EPROM), and Electronically Erasable Programmable Read Only Memory transistors (EEPROM)[6].

PLDs worked well for implementing simple logic functions but lacked the complexity to accommodate all ASIC type designs. A major drawback was that these devices lacked the ability to create latches, and thus could only be programmed for combinational logic.

During the mid 1980's a new technology emerged to fill the gap between PLDs, and ASIC technology. In 1985, Xilinx Corporation released the first commercial Field

Programmable Gate Array (FPGA)[6]. The initial capacity numbered only a few hundred gates, but has since increased to over 50,000 gate devices. This has given rise to the possibility of implementing many ASIC designs in a programmable device at a significant cost reduction.

Our initial interest in FPGAs for research was to exploit the reprogrammability of the chips. To date FPGAs have been used essentially as "glue logic" in most applications, taking the place of discrete logic chips, as well as in place of smaller ASICs. The exploitation of the reprogrammable nature of the FPGA has not been fully explored. One immediate application would be to use the FPGAs in an emulation environment. The initial focus of the research was to replace simulation by emulation, however most simulation technology has developed to the point that there might not be a significant advantage to justify switching to hardware. Consequently a simulation target was sought where the simulations were not running in an efficient manner. It was discovered that fault simulations involving feed back loops, could not be effectively simulated because the parallelism of the simulation was lost, due to the feedback loops. Thus, this area of emulation became our research target with the aim to build a fault emulation system that would exploit the reprogrammable nature of the FPGA architecture as much as possible.

1.2 Introduction to Field Programmable Gate Arrays

The concept of field programmable gate arrays has been in existence since the early 1970's[4] although it was not commercialized until 1985. The technology has slowly developed and is now a multi-million dollar industry. One of the main technological barriers to be overcome by companies manufacturing Field Programmable Gate Arrays, was the feature size. As the device feature size has shrunk, the ability to put more and more logic on a reasonable sized die has increased. In addition, as the feature size shrunk the

speed of the devices has also increased. These factors have lead the FPGA to be a commercially acceptable alternative to medium and large scale application specific integrated circuitry.

There are several large manufacturers in the FPGA industry today. The programming technology falls into four basic categories: static RAM based, anti-fuse based, EPROM, and EEPROM transistor based. Some of the major FPGA manufacturers include Xilinx, Actel, Altera, AT&T, and Atmel. A partial listing can be seen in Table 1.1.

Table 1.1 Major FPGA Manufacturers

Manufacturer	Primary Programming Technology
Xilinx	Static RAM
Actel	Anti-Fuse
Altera	EPROM & Static RAM
AT&T	Static RAM
Atmel	Static RAM
AMD	EEPROM
Cypress	Anti-Fuse
Algotronix[a]	Static RAM

a. Algotronix was bought by Xilinx, but is included in this list, as it was used for a major portion of the thesis research.

The anti-fuse based FPGA is an architecture that relies on a one-time programming sequence to configure the chip. The programming is achieved by sending high voltage pulses across a matrix of connections in order to connect two layers of routing, and

thus connecting associated logic and I/O. This type of FPGA architecture cannot be used in a reconfigurable system because it can only be programmed once.

EPROM technology can be reprogrammed, however, it is not easily reprogrammed in-circuit. To reprogram an EPROM device it must subjected to ultraviolet light, and then is usually programmed in a special programming device. For FPGAs using EEPROM programming technology, a significant amount of extra overhead would be needed to make the FPGA in-circuit reprogrammable, including additional voltage sources.

The static RAM based architecture can be split into two main categories; look up tables and multiplexer based. The look up table architecture is a large grained architecture that allows for a relatively large number of inputs (usually between four and ten), and one or two outputs. The basic design of a look up table cell is that of a small memory block. The logical output of each combination of inputs is stored in the table. The output from the table can then be fed out to a latch, or directly out to the next cell. The multiplexer based SRAM architecture uses the SRAM to control multiplexers, which then route the incoming signals to the appropriate logic gates. This architecture is primarily used with a fine grained FPGA design, (i.e. one consisting of 2 to 3 inputs and a single output).

Because the SRAM based FPGA is configured each time it is powered up, multiple uses of a single FPGAs are possible. It is also possible to reprogram the FPGA while in-circuit. This reconfigurability allows for the possibility of incorporating a reconfigurable part in a fixed system architecture, and it is this possibility that was the initial basis of this thesis.

1.3 Reconfigurable FPGA Based Systems

The concept of using a reconfigurable device in an otherwise non-reconfigurable system, as a reconfigurable part, has been looked at from a number of different angles. Some research has been done in using the reconfigurable parts in a stable system. This is

defined as a system in which the reconfigurable part does not change its configuration while the system is running. This type of system may be reconfigured and used for some new application at a later time. This is the most common type of reconfigurable system. Some examples of this type of system are as follows.

Quick Turn Technology has built a system around a large number of FPGAs and Field Programmable Inter-Connect (FPIC) chips. A large design (millions of gates or more) is then downloaded to these systems which act as an emulator. This type of emulation technology allows for fast prototyping of Very Large Scale Integrated Circuitry (VLSI) designs. Once it has been programmed, the system remains stable and unchanged until the next design is ready to be emulated. This is one example of a scalable multiple FPGA based system that may be fully reconfigured for multiple uses. Other examples include the BORG board, SPLASH boards, and the TM-1 (Transmogrifier) board, which are all based on the same principle and are designed for use in various applications[8][3][7].

Another approach to reconfigurable systems is to use a reconfigurable part in a stable system where the reconfigurable part may be reprogrammed while the rest of the system continues to run. There has been much less research in this area, but one company with some experience in this area is the Virtual Computer Corporation (VCC). VCC has produced a small board, based on an SBus interface, on which is mounted a Xilinx part and some RAM.[9] The concept behind this is to use the FPGA as a reconfigurable co-processor, and to be able to run certain time consuming algorithms in the hardware. There is a configuration for each algorithm, with the data stored in RAM. During an application the necessary configurations are downloaded as required. This is an example of an application in which the entire FPGA device is reconfigured while the rest of the system continues to run. This is a unique concept, not presently being implemented by many others.

There has been research into the area of combining FPGAs and other processors on the same die, and the general conclusion drawn is that the trade-off between area/speed, with the increased functionality and flexibility gained by the FPGA usage is a promising compromise[1][2][5].

Our initial research efforts focused on finding applications of dynamically reconfigurable systems. These were to be systems where the FPGA could be partially or fully reconfigured while the rest of the system was still in operation. This task was made more difficult because of the lack of availability of fully dynamically reconfigurable FPGAs.

1.4 Scope

The work done for this thesis focuses on two primary systems. The first system was built around an Algotronix CHS2X4 board, and the second system was built on an Aptix Programmable Circuit Board (PCB) using Xilinx FPGAs. The initial work, covered in Chapter Two, describes the design of a fault emulation system using the Algotronix 1024 FPGA. Using a manufacturer supplied board, fault emulation was accomplished by dynamically inserting faults into the FPGA platform. This approach fully exploited the dynamically reprogrammable nature of the Algotronix FPGA, and used a custom C based library of control commands to run the emulation process.

The second stage of this work moved on to using a non-dynamically reprogrammable device (the Xilinx 4010) to achieve the same goal as above. This involved a trade-off of hardware overhead to make up for the non-dynamic architecture of the FPGA. This platform was centered on an Aptix PCB using two Aptix 1024 Field Programmable Inter-Connect chips (FPIC). The control structure for this emulator was based around the HP 75000 test station. This is discussed in Chapter Three.

The purpose of both of these projects was to show the viability of using an FPGA based system as a fault emulation tool. If the emulator was capable of evaluating difficult to simulate test strategies, at the same speed as the easy to simulate test strategies, the emulator would be deemed successful.

Chapter Four presents general conclusions on the work done and possible recommendations for future work in both the area of FPGA based emulation as well as reconfigurable applications of FPGAs.

An appendix with an introduction to testing, and a glossary of the acronyms used in the thesis are included at the end of the document.

CHAPTER 2

EMULATION USING DYNAMIC RECONFIGURATION

2.1 Introduction

The original motive for this research was to find novel applications, that would exploit the reprogrammable nature of FPGAs. This was originally defined as applications that required dynamic reconfiguration. Algotronix produced an FPGA well suited for such applications. Algotronix also produced a PC bus based board with an array of up to nine of the CAL1024 chips. This board is known as the CHS2X4 Custom Computer. This product became the platform for our initial research.

It should be mentioned that after our research had started in this area, Algotronix was bought by Xilinx and no longer provided support for their products. This limited the amount of research carried out on this platform, and eventually led to the decision to try and build a Xilinx based emulation system.

2.2 Architecture of the CAL1024

The CAL 1024 is an SRAM based FPGA that uses a sea of gates architecture. The CAL1024 is an array of 32 by 32 cells, in which each cell contains interconnections to its four neighbouring cells, and a functional unit. The function unit allows for the implementation of twenty functions including all one and two input boolean functions, latches, and constant output (zero or one) functions. The CAL 1024 is considered a fine grained FPGA,

as all the functions have at most two inputs, and one output. The general structure of the array is shown in Figure 2.1. The functional unit is shown in Figure 2.2.

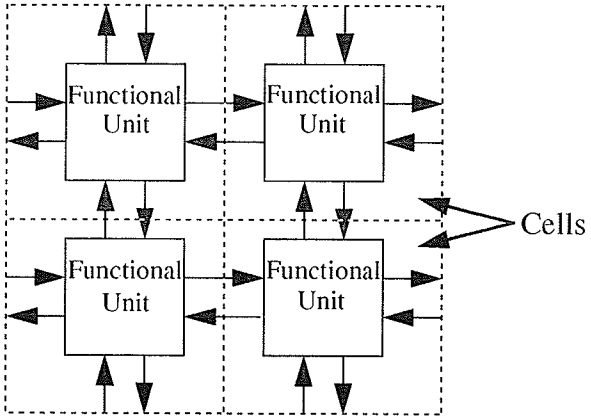

***Figure 2.1** CAL1024 Array Structure*

The functional units and the cell to cell routing are controlled by static RAM cells called control store RAM. There are no routing channels with this architecture, and because of this the routing resources reduce the number of function units available. That is to say, if all the routing resources of a cell are being used for general routing purposes the

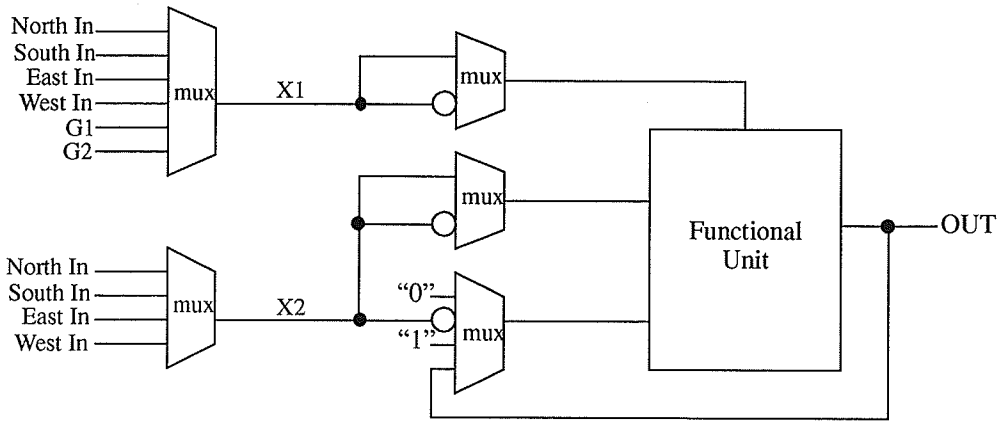

***Figure 2.2**Algotronix Functional Unit*

functional unit of that cell becomes unusable, as there is no way to input or output signals from that functional unit. This was one of the major limitations of this technology.

The control store elements may be accessed by the user, and individual blocks of RAM can be reset by down-loading the new control information to those blocks only. This enables partial reconfiguration on the cell level, a feature exploited by the emulator. The actual manipulation of cell data was controlled by a set of C routines provided by the manufacturer.

Another very useful feature of the CAL1024 control store RAM was the inclusion of one bit, that contained the present state of the output of the functional unit. The ability to read back the state of any functional unit meant complete observability of the circuit without having to dedicate extra lines or pads for observation purposes. Because the I/O were not needed to observe the state of the circuit, the "output" from the emulator, could be placed in any physical location on the CHS2X4 board. This feature allows for more flexibility in the floor-planning stage of the design flow.

The cellular structure of the CAL1024 allows for transparent boundaries when cascading the chips in an array. The I/O have a unique pad sharing scheme which uses a three level logic scheme, that allows an input and output signal to share the same pad[15]. This feature reduces the I/O count, and simplifies the cascaded setup.

2.3 The CHS2X4 Board

The CHS2X4 consists of an array of up to nine CAL1024 chips, of which eight are available to the user while one is dedicated for control purposes. The board has a PC/AT bus based interface through which all communications to the board are handled. The board also has set of RAM banks, which were not utilized for this project. All communications with the board where handled through a set of manufacturer supplied C routines.

This enables the user to manipulate such tasks as cell output read back, routing manipulation, and most importantly for this research, cell functionality.

When generating a design for the emulation, standard schematic capture tools could be used, or the designer could hand place and route the design using the Algotronix Configurable Logic Array Editor (CLARE). After a number of design attempts it was realized that CLARE was the only way to place and route design of any practical size.

The single CHS2X4 board had a maximum capacity of 16,384 gates. The usable capacity of the system was much less, as many of the functional units could not be used in cells where the routing resources were already being used. This limited the size of circuit emulated, but was not the major constraining factor. Instead the software used to compile the designs for emulator was the major constraining factor. It could not compile designs of this size due to memory limitations.

2.4 Emulation Methodology for Stuck at Faults

A variety of emulation methodologies have been investigated and will be discussed. They are all based on the following principles. The circuit under test is first loaded onto the emulator for testing. The faults can then be dynamically injected on an individual basis. The circuit is tested one fault at a time. Multiple faults are not considered. From these principles three emulation methods have evolved.

The first method is sequential emulation, the basis of which is as follows. The golden (fault free) circuit downloaded to the emulator. This circuit is then clocked for the duration of the test cycle, for illustration say 10,000 cycles. The signature is then recorded and the circuit is reset. A fault model is now loaded into the circuit, and is again clocked for the duration of the test. The new signature is recorded and stored for analysis. This continues, until all of the faults have been checked. This methodology maximizes the available area for the circuit under test, but is also the slowest method for fault grading, as

faults are not dropped once they are detected. However, this is the fastest method when the entire test cycle must exercised, as the emulation control software does not interrupt the normal clocking cycles to check signatures, or compare bits during the test cycle. The procedure is illustrated in Figure 2.3.

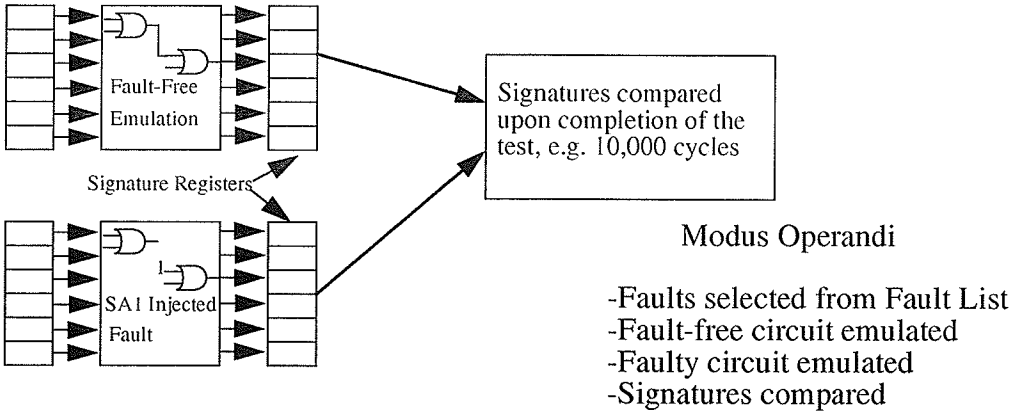

***Figure 2.3** Sequential Emulation with Single Signatures*

The second method, shown in Figure 2.4, still allows for maximum circuit size, but reduces the testing time by reading multiple signatures. The golden circuit signatures are initially recorded at some set intervals. When the faulty circuit is tested, the test pauses at the given interval and analyses the signature off line. If the signatures do not match the fault can be dropped and a new fault inserted. If the signatures do match, the test continues through the next interval and the process is repeated. The testing time will be reduced by a factor, dependant on how early each fault was dropped. The off board comparison time is negligible in comparison to the total test time. In this case the use of the C routines to control the board functions is very advantageous as the compare routines can be written into the same program that controls the emulation. In this way the host processor is used to supplement the emulator hardware, and becomes a part of the emulation system. This is functionally opposed to most applications of FPGAs used for custom computing

machines. In most applications the custom computer is used to augment the calculation done on the host processor, not the other way around

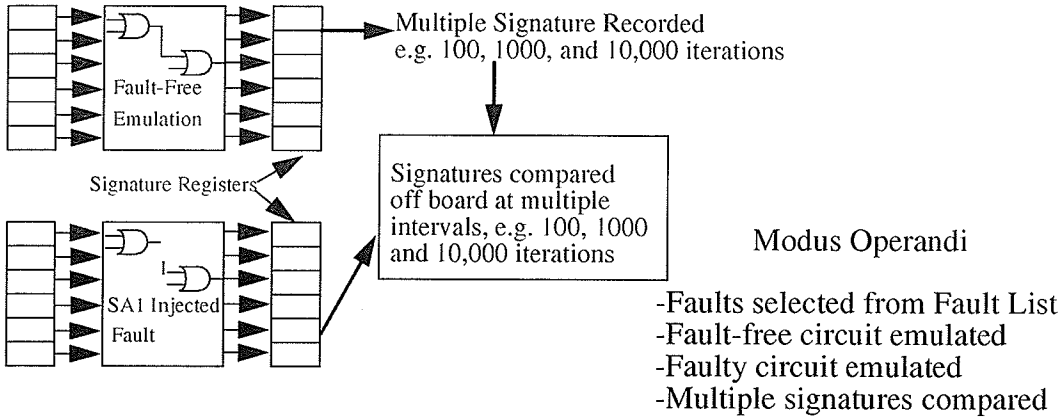

Figure 2.4 *Sequential Emulation with Multiple Signatures*

The third method, denoted as simultaneous emulation and shown in Figure 2.5, should be the fastest, however, it trades off speed for allowable circuit size. With this method, two circuits are loaded to the emulator. One is considered the golden circuit and the other will be designated the faulty circuit. The outputs from the two circuits are wired to a comparator. The faults are injected individually as in the previous schemes. As the circuits are clocked the output of the comparator is checked once every clock cycle. As soon as the comparator signal goes high, it signals that the fault has been detected, and the test is halted. The circuit is then reset, and a new fault injected, and the process is repeated. The test of the comparator output is controlled through the same software program that controls the other emulator functions.

Of the three methodologies presented, only methods one and three are really needed. For fault grading, and detectability profiles, method three is preferred. To test for

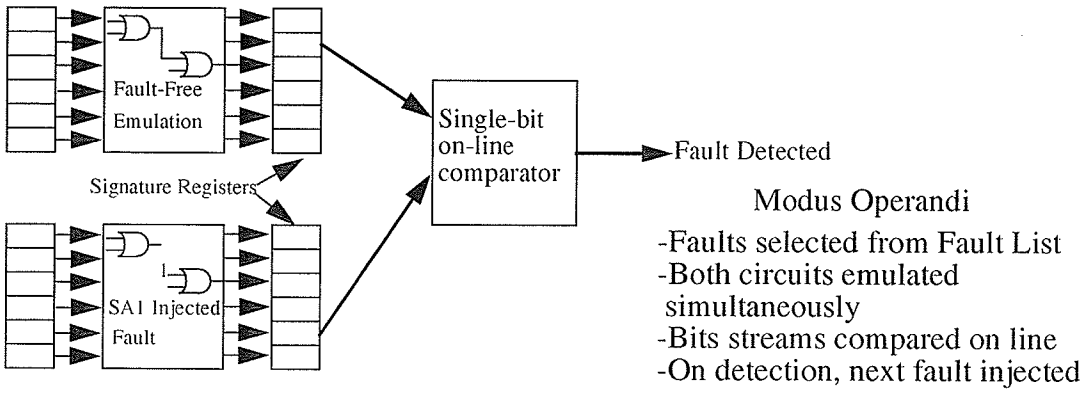

Figure 2.5 Simultaneous Emulation

aliasing method one is preferred as it is necessary to clock the circuit to the completion of the test cycle, and method one is the fastest method to achieve this.

2.5 Fault Grading, and Detectability

The 74LS81, a four bit ALU[16], was chosen as the initial circuit used to test the emulator. The 74LS181 is a combinational circuit that has been widely used as a bench-mark for various test methodologies[17]. The procedures used in this research can be extended to larger and more complex circuits, however, the physical limitations with the Algotronix software prevented the design from increasing in size. This size restraint is the main factor that lead to work on the Xilinx based emulation system.

Initially when testing the 74LS181, the simultaneous emulation method (method 3) was used. With both circuits, the golden and faulty circuit, the emulation system was very close to its maximum capacity.

The first task of the emulator was to fault grade the circuit. This involves injecting faults, and observing whether the fault propagates to the output. This does not involve using a signature compaction scheme. The purpose of this test is to merely find out if a

fault's behaviour is observable at the output, whereas the detectability is a measure of how often or how easily the faulty behaviour manifests itself at the output. The test vector generation for both the fault grading and detectability tests, was provided by a maximal length 16 bit LFSR. From the LFSR fourteen bits were tapped and fed to the inputs of the circuit. This provided for a maximum of 2^{14}-1 possible input vectors.

The fault list for the ALU consisted of 314 equivalent stuck-at faults. Because the Algotronix FPGA allows for a maximum of two inputs per gate, three or greater input gates were implemented with the appropriate cascaded two input gate equivalents. This does not effect fault coverage, as the 74LS181 is fully observable. When testing circuits with an undetermined or less than 100% observability, faults arising from the cascaded version of multiple input gates could be taken into consideration when defining the fault grading and detectability profiles in order to reflect the actual statistics of the original model. A one to one mapping between the original circuits and emulation implementations can be easily generated and this will ensure that only faults which correspond to the original model are taken into consideration.

Fault grading of the ALU resulted in all of the faults being observed within about 350 cycles for several different seeds. Once the fault has been observed, the circuit is reset and the next fault is injected. Using this process and running the computer at a reduced bus speed of 8 MHz, the entire circuit was fault graded in approximately one minute. This process could take longer if for instance there were faults that were harder to detect. However, for a given set of faults the procedure will run in the same amount of time regardless of the complexity of the circuit. As well, as the circuit increases in size the time needed to fault grade will only increase at a rate based on the size of the fault set, not the circuit size itself. Emulation time is dependent on the delay associated with the critical paths in the circuit which may be dependant on circuit size. It can be said however that the emulation

time is essentially independent of circuit size. This is a major advantage of emulation systems and is not the case with simulators.

The detectability of the faults was the next task performed. This test was implemented in the following manner. Individual faults were injected into the circuit under test. The circuit was then clocked for 2^{14} cycles to insure the majority of possible input pattern were covered. The outputs from both the golden and faulty circuits were used as inputs to a comparator circuit. Each time the comparator observed a fault, it was recorded. The detectability was then calculated as follows:

$$D = \frac{O_F}{C_T} \tag{2.1}$$

where D is defined as the detectability of the fault, O_F is the number of times the fault is observed and C_T is the total number of cycles. The detectability profile of the 74LS181 can be seen in Figure 2.6.

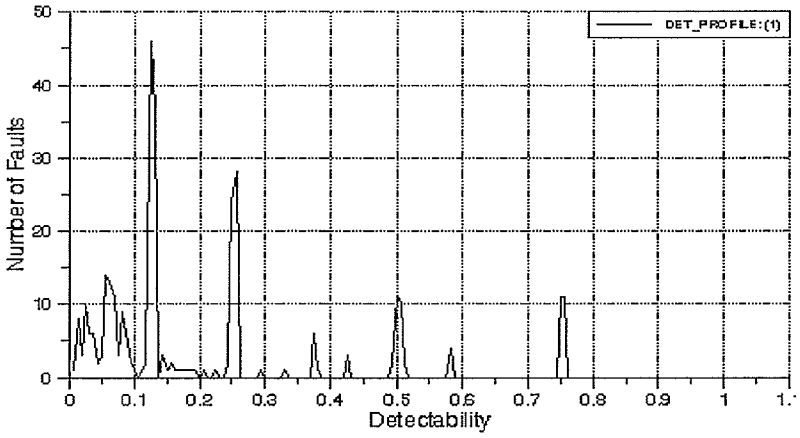

***Figure 2.6** Detectability Profile of the 74LS181*

2.6 Stuck-at Fault Emulation Results

The initial BIST scheme implemented was the LFSR scheme. This is a standard structure for most BIST implementations. When testing the LFSR scheme, only method one and three were employed. Method three, simultaneous emulation with multiple signatures, was used to quickly observe whether all the faults in the circuit were detected by the LFSR. It was expected that the LFSR would detect all the faults, as the 74LS181 is 100% fault observable. The initial emulation results showed that the LFSR did detect all faults.

The second test assigned to the emulator was to check for aliasing of the LFSR. This was accomplished using method one, sequential emulation using a single signature. Once it was known that the circuit was 100% detectable by the LFSR, it had to be ascertained whether any of the signatures would alias. To establish this the circuit was clocked to the end of the normal test cycle, and the signature analysed. As long as the signature did not match the good signature, aliasing had not occurred. Each fault was analysed and it was found that no fault caused the LFSR to alias.

Circular Self-Test Path (CSTP) was the second scheme to be implemented on the emulator. The configuration of the test path is shown in Figure 2.7. From Figure 2.7 it can be seen that four of the primary outputs are fed back to four of the primary inputs (F0-F3 and B0-B3 respectively) as "active inputs". The other primary inputs are set to a "0" value as "non-active inputs" to form the ALU CSTP test configuration.

As mentioned earlier, the signature for a CSTP scheme can be taken as a combination of the content of any of the registers in the test path, or as a single bit stream from one of the registers in the test path. For our tests all the registers in the test path were used to analyse the signature. This allows for a signature length of 22 (14 inputs, and 8 outputs).

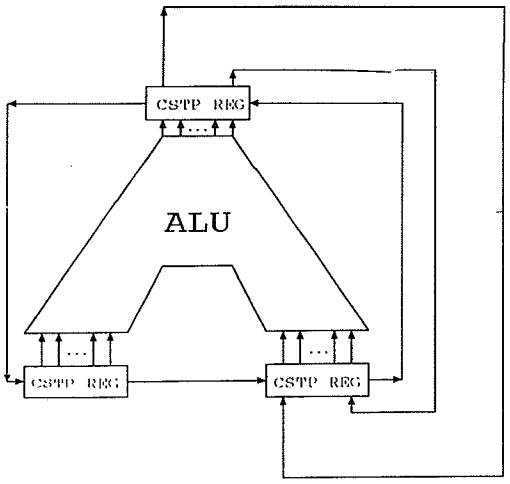

Figure 2.7 Four Bit ALU CSTP Configuration

As with the LFSR scheme, all the faults were detected and no aliasing was found to be occurring.

As the CSTP scheme has not been widely tested, it was decided to augment the previous tests with other tests that would explore some of the statement made in [18]. The first of these tests was to examine the probability of any of any bit remaining in a single state. Krasniewski and Pilarski have stated that after a short period, the probability of any bit being set to one should approach 0.5. This was verified on the emulator, as the probability of one on all the inputs diverged to 0.5 with an actual count of 0.49657 over 10,000 cycles.

The other property of CSTP examined was state-space coverage. The state-space coverage was implemented for three randomly chosen eight bit input taps, of which the chosen eight bits included a hybrid of "inactive inputs" and "active inputs". The state coverage at different cycle lengths are given in Table 2.2. Three randomly selected eight bit words for this experiment are denoted Tap 1, Tap 2, and Tap 3. We noted that after 10,000 cycles nearly 100% of the states are generated on all three taps. This verifies that the

CSTP registers will cover most possible states in a reasonable number of clock cycles. This is essential for any test pattern generation scheme, and these findings show that CSTP appears to be a viable alternative to LFSR based test generators.

Table 2.2 State Coverage of Inputs

Cycles	Tap 1	Tap 2	Tap3	Average
100	37.81%	33.20%	32.03%	34.35%
1000	97.27%	96.88%	96.48%	96.88%
10,000	100.00%	99.61%	99.61%	99.74%

It is important to note that the test performed on the CSTP scheme took the same amount of time as the test performed on the LFSR scheme. This should not be the case if a simulator was used. The ability to develop an verify different BIST strategies such as CSTP, will not only help to increase the new BIST techniques acceptance, but will also decrease the time for a system to be validated. Although the CUT being used was a very basic design, this methodology can easily be extended to larger and more complex designs.

2.7 Delay Fault Testing

Transition delay faults are defined as faults caused by slow transitions occurring on a line on the circuit. Because the state of the line is either slow to rise or slow to fall the state of the line may at times have erroneous values when the registers in the circuit are clocked. In many ways delay faults appear to manifest themselves as a stuck-at faults. Differences include the fact transition faults only manifest themselves when the value of the line changes state. For example a slow to rise fault may only manifest itself for one clock

cycle if the state of the line is "0, 1, 1, 1" whereas a stuck-at-zero fault would be manifested for three of the four cycle.

Several paradigms for implementing the transition delay fault were explored. The first example can be seen in Figure 2.8. The proposed model consists of three similar circuits. The first circuit is designated the golden circuit. The second circuit is the circuit in which the stuck-at faults are injected on. On the third circuit is a golden circuit, that is delayed by one clock cycle. When a fault is observed at the output of the faulty circuit, the delayed circuit is probed to find out the delayed value of the line under test. If the delayed circuit test point and the test point on the faulty circuit have the same value, the delay fault is observable, and should be detectable with most testing schemes.

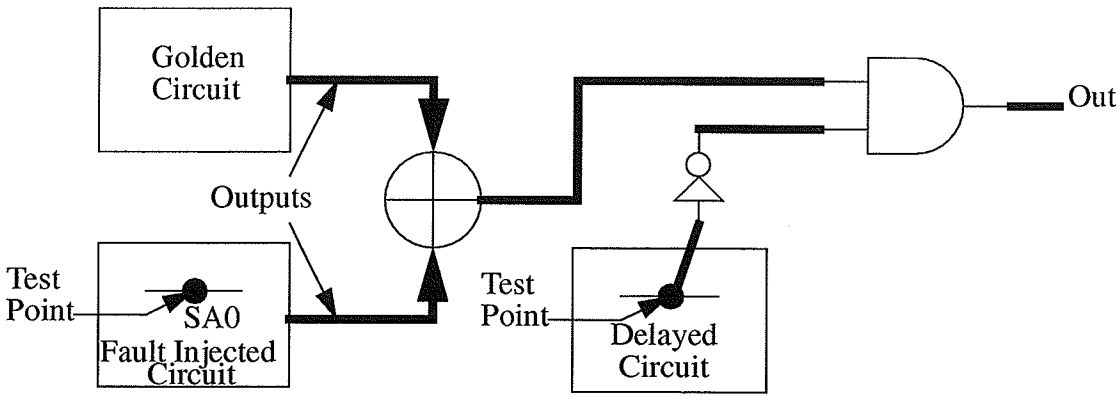

***Figure 2.8** Delay Fault Paradigm #1, for a Slow to Rise Transition Delay Fault*

Implementation of the above paradigm was attempted. Unfortunately the Algotronix software could not compile a design of this size, due to PC memory constraints. Because of this set back a second delay fault paradigm was defined.

The second method, shown in Figure 2.9, is less hardware intensive and is implemented by augmenting the golden circuit with flip-flops. The flip-flop acts a shadow regis-

ter of the previous state of the line connected to it. Because of the routing constraints of the emulator, not all of the delay faults were tested. Instead the lines with the 10 least detectable stuck-at-zero and the ten least detectable stuck-at-one faults, were chosen to evaluate the corresponding ten slow to rise and ten slow to fall faults.

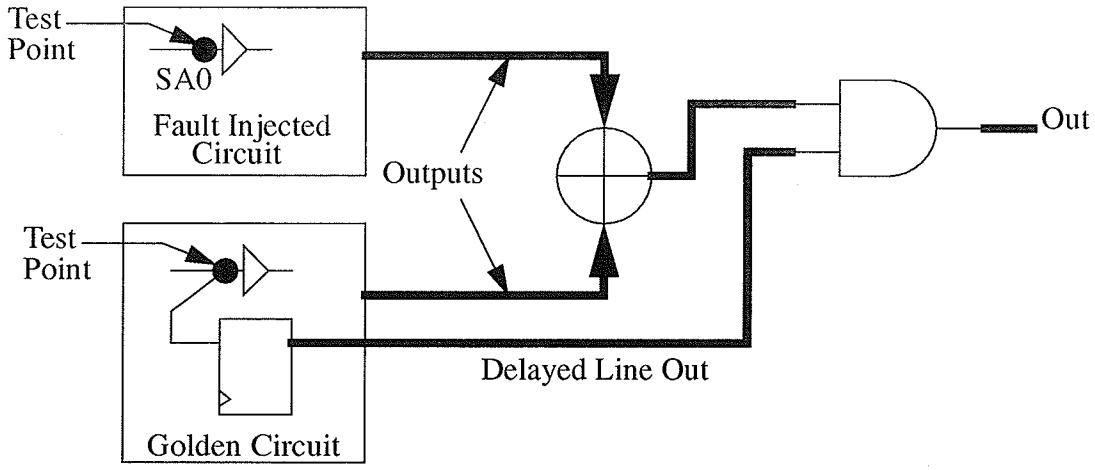

Figure 2.9 *Delay Fault Paradigm #2, for a Slow to Fall Transition Delay Fault*

Due to the similar nature of delay and stuck-at faults, the following approach was used to evaluate the delay faults. When evaluating a slow to fall fault, a stuck-at-one fault must be used in the fault injected circuit in order to determine the detectability. Likewise a slow to rise fault uses a stuck-at-zero fault to test for detectability. If the stuck-at-zero fault of line l is detected there is a probability (determined by the zero controllability of line l) that the slow to rise fault will also be detected, as the slow to rise fault must be in a low state during the previous clock cycle if it is to be detected.

From the test it was determined that all of the transition delay faults tested were observable. The detectability of the faults were not calculated, but could be examined with some minor changes to the circuitry.

The second paradigm is sufficiently efficient for testing most delay faults, as it should only be necessary to test delay faults where the stuck-at fault has a low detectability, since the transition faults corresponding to those stuck-at faults have a lower detectability [19].

2.8 Algotronix Based Emulator: Rants and Raves

The system described above, is by no means a polished commercial product. It does address the question of whether FPGA technology is a viable platform on which to base a fault emulation system. The answer to this question is yes. In this section the shortcomings, and assets of an emulation system using Algotronix FPGAs will be discussed.

The first issue to address is the speed of the system. Although the emulator time was relative fast, it could be greatly improved by changing the clocking system. The clock is generated by commands issue in the C routines used to control the board. This means that the maximum frequency of the system is 4 MHz. However the potential speed of the emulator, although being design dependant, is faster with the design currently being implemented. A clock on board the CHS2X4 or on chip should increase clocking speed by avoiding the PC bus bottleneck.

The architecture of the CAL 1024, is not entirely perfect for implementing designs. Specifically the sea of gates approach, although excellent for local communications, and inter-chip communication, is very difficult to implement for bus based designs. The addition of routing channel architecture to such a chip would be invaluable. The greater routing resources would mean both increased utilization of the functional units as well as the ability to implement larger, and more complex designs in a smaller physical area.

One of the advantages of the CAL architecture, is its fine grained implementation. Although fined grained FPGAs are currently in disfavour as a general FPGA architecture, they are well suited for emulation based applications. Fine grained architecture makes it possible to manipulated the design a gate level with the greatest of simplicity. For instance, even if it was possible to reconfigure a Xilinx part at the look up table level, the user would still have to deal with up to an eight input function that may be implementing the logic for a number of gates in a design. With the fine grained architecture, each functional unit corresponds to a single gate on the emulated design. This one to one mapping is a great advantage when attempting to extract information from the emulator.

The other superb feature of the CAL architecture is the single bit of control store RAM used to store the current state of the outputs of the functional units. This feature allows complete observability of the FPGA with a minimum of I/O pin overhead. This feature is found only on the CAL FPGA and is ideally suited for emulation technology. To implement this level of observability on any commercial FPGA would involve dedicating an output pin for each FPGA or capturing the outputs, in registers and shifting them out. Either scheme is impractical as the largest limitations of FPGAs include register implementation (usually one per logic unit) and pin count (there are always more logic units than pins on an FPGA).

2.9 Chapter Summary

In this chapter a unique fault emulation scheme based on an Algotronix CHS2X4 board based platform has been presented. The emulation uses the advantages of fine grained FPGAs to fault grade, and test BIST schemes for both stuck-at faults and transition delay faults.

The tests performed using the emulator have shown that CSTP is an effective BIST strategy, and that CSTP and other BIST schemes can be examined with equal efficiency, regardless of the BIST circuit complexity.

CHAPTER 3

STATIC RECONFIGURABLE EMULATION

3.1 Introduction

Due to limitations of the Algotronix based emulator discussed in Chapter 2, a new emulation platform was developed. This emulator relied on the scalability of reconfigurable hardware, instead of the need for dynamic reconfiguration as a basis for the emulation process.

This second generation emulator is based on an Aptix Field Programmable Circuit Board (FPCB), which includes two Aptix 1024R Field Programmable Interconnect Devices (FPIDs). Four Xilinx XC4010PG191 FPGAs are mounted on the FPBC. The emulator is controlled off-board by a HP VXI test station. The complete emulator configuration is shown in Figure 3.1. Included in this figure is the Aptix G2 FPCB and the hardware and software needed to interface to the data I/O and control lines. A photograph of the emulator board is shown in Figure 3.2.

3.2 Xilinx Architecture

The Xilinx 4010 FPGA is an SRAM based FPGA. It contains 400 Configurable Logic Blocks (CLBs) arranged in a 20 by 20 array formation. The FPGA is capable of

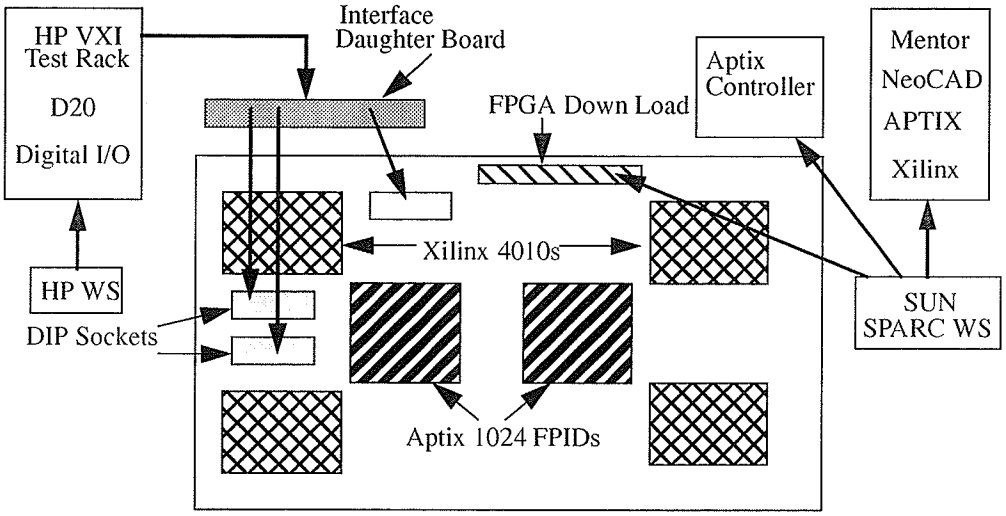

Figure 3.1 *Emulator with Hardware and Software Interface*

implementing approximately 10,000 gates. The FPGA routing architecture is implemented through routing channels, connected by an array of switch matrices.

The 4000 series FPGA logic architecture is based on Configurable Logic Blocks (CLBs) consisting of two four input Lookup Tables (LUTs) and one three input LUT as shown in Figure 3.3. The CLB is capable of implementing any two independent four input functions, any five input function and some functions of up to nine inputs. The CLB con-

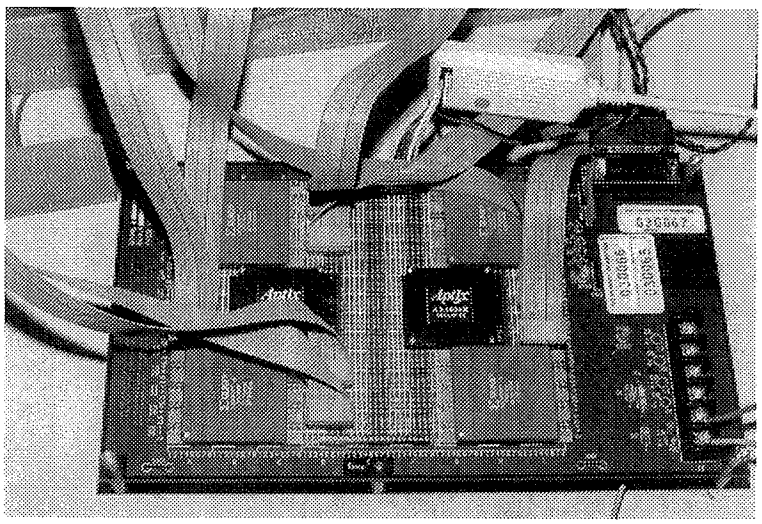

Figure 3.2 *Photograph of the Emulator Board*

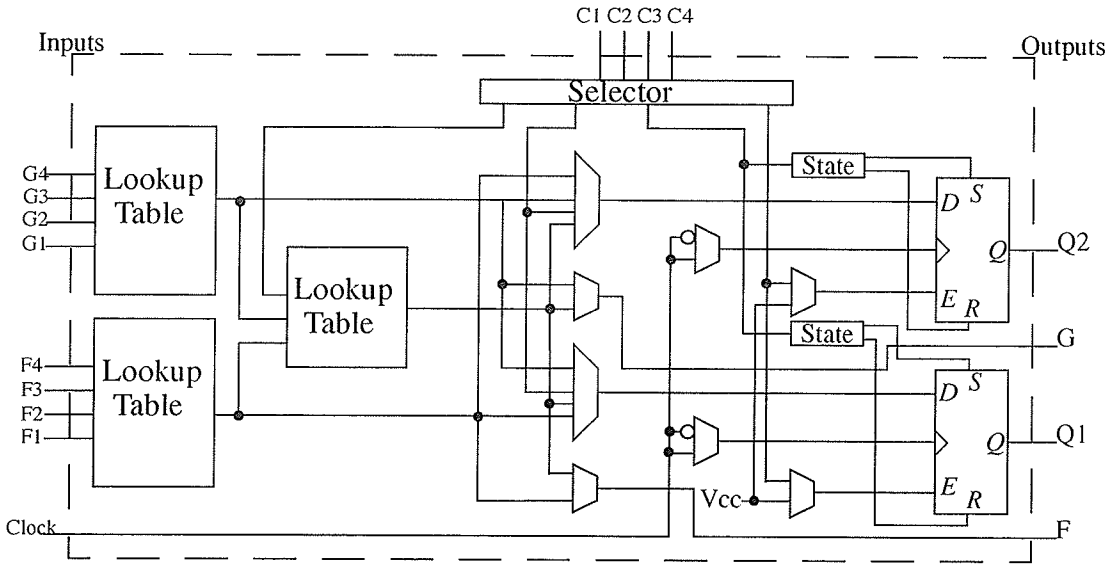

Figure 3.3 *Architecture of the Xilinx 4000 Series CLB*

tains two latches, and thus can latch any of the outputs from either the four input LUTs or the three input LUT[6][20].

The Xilinx FPGA cannot be partially reconfigured. It is reconfigured through a single bit-stream, which is stored in the configuration RAM. The inability to partially reconfigure the device was the reason why the Xilinx FPGA was not initially chosen as a platform for emulation. Due to the large capacity of the Xilinx architecture and the tools available, the FPGA can be configured to such an extent that the dynamically reconfigurable elements missing from the FPGA architecture can be added as overhead to the circuit being emulated.

The XC4010 has 160 input/output blocks (IOBs). The reasonable high IOB count was one of the reasons the XC4010 was chosen. When partitioning designs over multiple chips, one of the key constraints becomes the availability of I/O pins.

3.3 The Aptix Field Programmable Circuit Board

The Aptix FPCB is a multi-layer circuit board on which is situated a dual in-line pin array, consisting of 1552 pins. On the board are two 1024R FPIDs, each capable of providing 1024 interconnections. The board is divided into two regions, with one of the FPIDs in each region. There are also a number of global signals that can cross the boundary between the two regions.

The FPCB, although reprogrammable, is not easily reconfigurable in terms of the hardware placed on the board. This is due to a number of factors. First, because the board uses a dual in-line package (DIP) type of pin array, adapters are needed to place any of the Xilinx FPGAs (with the exception of the XC2064) on the board. For the XC4010 FPGAs used in the emulator, this meant inserting both a 192 dual in-line pin adapter on the Aptix board, and then inserting a 191 pin grid array FPGA onto the adapter. This is not an easy process to accomplish manually. In this case a drill press, two pieces of soft compressible styrofoam, and a great amount of care were used to accomplish the task. Once set in place it was expected that the FPGAs would not be removed in the near future. The second constraint of Aptix board, is that power and ground pins cannot be routed through the FPIDs. To facilitate the wiring of the power and ground connections a power plane is available to all pins on the top layer of the board, and a ground plane is available to all pins on the bottom layer of the board. To connect the power or ground planes, zero ohm resistors are surface mounted between the power or ground plane and the associated pin. This means that the placement of devices on the board, and the associated power and ground pins must be known when initially planning a layout on the board. If the design is changed, the zero ohm resisters used previously must be removed, and new ones soldered in place. This further complicates rearrangement of devices on the FPCB.

Once the board configuration was decided upon, the Aptix board was well suited for our emulator application. The final configuration included four XC4010PG191 FPGAs, eight DIP sockets for interfacing with the Hewlett Packard test station, and four sets of single in-line pins (SIPs) to attach to the Xilinx X-Checker cable used to program the FPGAs. The FPGAs could have been configured to be connected in a daisy-chain configuration for programming, and in this fashion there would only be one set of pins needed for the X-Checker cable. However, to allow more flexibility in the pin configuration it was decided to have a separate set of pins for programming each FPGA. In this manner the pins used for programming were unrestricted when also used for I/O. The final configuration can be seen in Figure 3.4

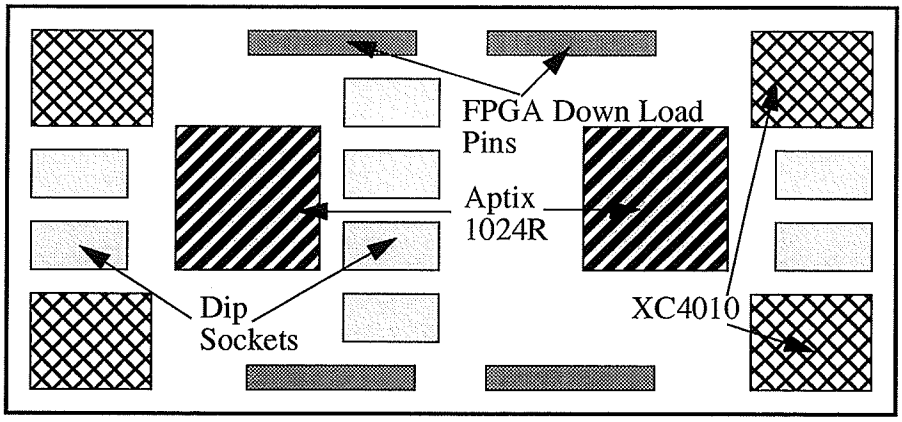

***Figure 3.4** Layout of the APTIX FPCB*

3.4 Emulator Control Environment

A Hewlett Packard workstation, in conjunction with a HP VXI test station, were used to control the emulator. The workstation was running HP D20 test software. The D20

spreadsheet environment was used to generate all control signals for the emulator. The D20 software also served as the data storage and evaluation facility.

The one function that could not be generated through the D20 software and VXI test hardware, was the system clock. The software allows for a system clock, but the clock must be generated at the end of each test cycle. In the configuration for the emulator a test cycle was defined to be the entire clocking period. If this definition had been changed to meet the software requirements, the software still would not have been capable of running the emulator, as it did not have the memory requirements for our tests (65,536 clock cycles multiplied by 1964 tests, or approximately 129 million states). Instead the system clock was generated by the oscillator on one of the Xilinx FPGAs. At the end of a test cycle a trigger signal was generated from the FPGA and the tester disabled the clock until the next test cycle was ready to begin.

3.5 Design Software

The designs were initially entered using Mentor Design Architect and a set of custom Xilinx based libraries. Once a design was completed it was saved as an XNF file. The XNF file is an intermediate Xilinx design file. This file contains all the design information but the design has not yet been mapped to the target technology. The XNF files are then translated to a NeoCAD format. NeoCAD is the primary tool used to map, partition, place and route the design.

3.5.1 Mapping the Design

The initial design used 1342 CLBs partition to four XC4010 devices. Although the NeoCAD design tools had worked quite well on smaller designs, it quickly became apparent that for designs of this size some manipulation of the design flow was needed. The initial step in the design flow was the translation and mapping of the XNF file to an NCD

NeoCAD format. This was handled without any difficulties by the NeoCAD tools. When this process was complete the resulting file contained an over-mapped design file. This is a file where the logic and connections have been mapped to fit into the 4000 series CLBs. Also included in this design file was new connectivity information. Many of the original logic functions will have been combined to fit into one logic block and thus the connectivity information from the original design file, will differ from the connectivity between the CLBs. Over-mapping refers to the fact that there are more CLBs (1342) than will fit into the target FPGA (the 4010 can hold 400 CLBs). It should be noted that at this point the only information in the NCD files are the contents of the CLBs and the connectivity information between the CLBs. An illustration of the mapping process is shown in Figure 3.5.

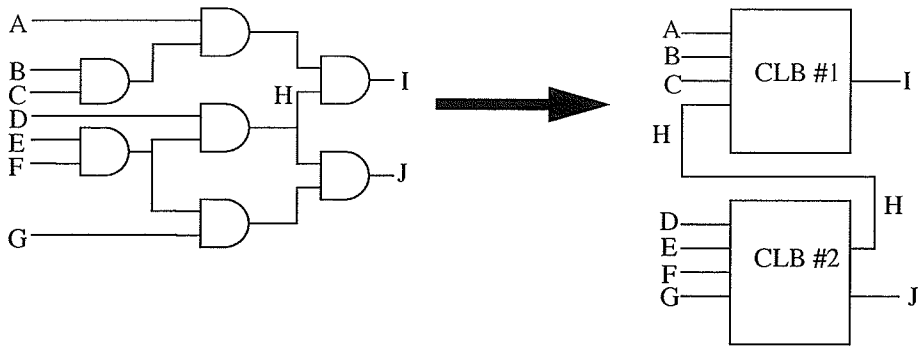

Figure 3.5 Generic Mapping of Logic to CLBs

3.5.2 Partitioning the Design

The next step in the design process is to partition the design to multiple FPGAs. In this step the NeoCAD partitioning tool known as PRISM, divides the original design file into four smaller design files, that are interconnected. This process must evaluate several different factors when deciding how to implement the partitions.

Speed and timing factors can be critically affected if the partitioning is done in an inefficient manner. For example, if a path in the circuit, defined as the logic between two registers, passes through four gates the partitioning software should keep this group of logic together on one chip if the speed of the design is to be kept at a maximum. If the four gates are partitioned to four separate chips, the delay associated with this path will increase not only from the delays associated between FPGA interconnections but also the I/O pad delays and the internal routing between the I/0 pads and the logic block on each chip.

Insuring that register to register paths remain on a single chip is also crucial when evaluating the operating frequency of the final design. Currently timing specification are only available on a chip by chip basis. Thus if a worst case maximum operating frequency of 5 MHz is calculated by the Xilinx static timing analyser, it means that worst case operating frequency on a single FPGA is 5 MHz. If however, a path in the overall design is split between the four chips, as mentioned above, the combined delay could very well decrease the operating frequency of the emulator. For large designs, this type of information is not currently available from PRISM

Prism does have the ability to perform timing driven partitioning. This can be used to specify a minimum operating frequency and PRISM, using worst case delay scenarios will partition the design to meet that frequency. Unfortunately, this feature does not work for large designs such as those implemented here.

When a design is partitioned, the partitioning process defines new inter-chip connections. These connections are allocated I/O pins on each of the individual FPGAs and each new set of connections shares the same I/O name. The Prism software can evaluate the new netlists, and create a file listing all interconnections between the various FPGAs.

3.5.3 Place and Route

NeoCAD placed and routed each of the individual designs for the emulator. Placement is defined as allocating each of the CLBs to a physical location on the target FPGA. Routing is defined as allocating each connection or net to a physical routing channel on the FPGA. The most critical element determined by the place and route process is the on chip delays. These delays ultimately define the frequency of the emulator. To guide the place and route process, critical nets and timing constraints can be specified. NeoCAD will attempt to meet the timing constraints, and continues the place and route process, scoring each attempt based on the user defined constraints. In this way, even if the design does not meet the original timing criterion, the best attempt is clearly identified.

Once the place and route process is complete, the design is translated to a Xilinx LCA format. This step is taken for two reasons. Firstly the download cable provided by NeoCAD is not currently in working order. Because of this a Xilinx download cable is used and this cable uses a bit stream that is generated from the Xilinx makebits program. Secondly, timing information could not be extracted for the design being implemented using the NeoCAD timing analysis tools, and the only way to extract the timing data for each of the FPGAs was to use the Xilinx static timing analyser. When the LCA file has been created, the bit file was generated and the FPGA designs were individually downloaded.

3.5.4 Scicard Netlist

After the partitioned designs have been placed and routed, they are used to create a scicard netlist. This netlist, generated by Prism, contains information such definitions of the parts in the design, and the connections between each part. The netlist generated by Prism is the basis for the final emulator netlist. At present other design information must be entered by hand. In the future much of this can be automated. The additional informa-

tion includes power and ground nets, and connections to the control DIPs, the data retrieval DIPs and the connections to the download SIPs.

3.6 Emulation Methodology

The previous emulation methodology, described in Chapter 2, relied on dynamically switching in and out fault models. Because the dynamic reconfigurable technology originally used is no longer in production it was decided to attempt to build a fault emulator platform around a non-dynamic technology. This meant re-evaluating the fault insertion methodology.

Dynamically reprogramming the entire emulator was a possible method for executing the fault insertion. This would mean compiling a new bit stream for each fault model in the design. This process could be accomplished in two ways. One would be to automatically produce a new netlist for each fault, by parsing the existing netlist, and extracting the necessary design information. For each netlist a new bit stream would have to be created, and downloaded to the emulator.

The second method would involve manipulating the bit file, to change only the bits needed to implement the fault models. Unfortunately this method would also involve a great deal of reverse engineering, as the bit stream configuration is not public domain.

Due to the complexity of creating all the netlists, and the time involved in programming the emulator, dynamic reprogramming was not considered a reasonable alternative.

The alternative to dynamic reprogramming and dynamic reconfiguration was static reconfiguration. This is defined as a design that has built-in reconfigurable hardware, but where the FPGA does not require reconfiguration. Essentially all the reconfiguration needed was built into the emulated design. The fault models are programmed into the circuit at the design stage, and can be switched in or out through the use of multiplexers con-

trolled off-board by the VXI test station. This design methodology trades off the hardware resources of the FPGA with the speed needed to switch the multiplexed signal. For instance, if dynamic reprogramming was used, the FPGA resources used would be less. At the same time, the reprogramming of the emulator takes a considerably greater amount of time than the switching of a single multiplexer.

Each "fault gate" consists of not only the regular gate in the original design but also all the fault models for that gate and multiplexer hardware to switch in and out the fault models. All the extra hardware is added at the design stage, through the use of a custom Xilinx library. For example, a simple AND gate is represented in the fault library by all four of its fault models as well as the AND gate itself. This is shown in Figure 3.6.

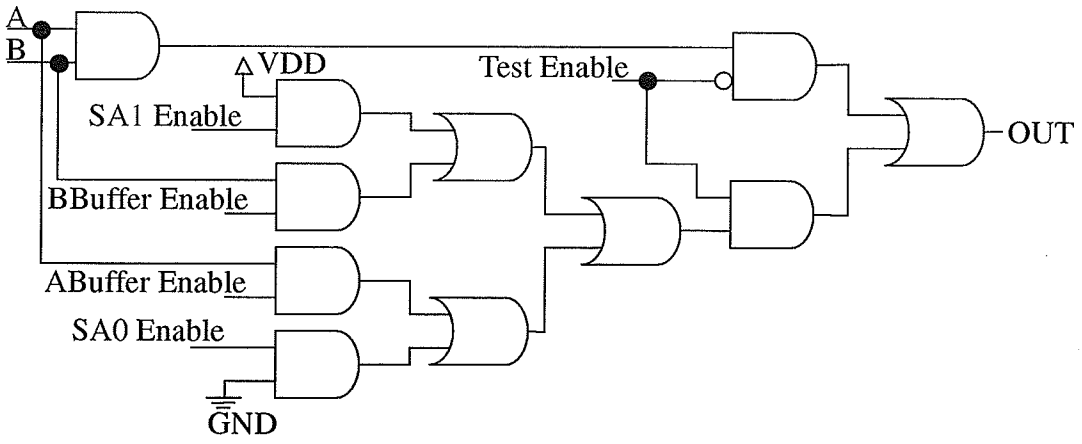

***Figure 3.6** Fault Emulator AND Gate Representation, With Fault Models*

On average for the designs now being implementation there is approximately a 1:12 gate ratio between a normal design and a design using the fault models. This ratio includes the fault models and control overhead. Every gate used in the design must also have a dedicated signal in order to put that gate into the TEST mode or the NORMAL

mode. This signal must be decoded on the chip to save both pin count and to keep the I/O bandwidth of the controller to a reasonable level.

The current configuration of the VXI tester allows the user to control and generate 64 I/O signals, as well as another eight control signals. The definition for a control signal, in terms of the tester, is that it is a single signal which can only be configured as an output from the test station. Table 3.3 shows a complete listing of signals and their designated tester modules. Fault_Control[0-5] refers to the signals that control which type of fault model is currently being accessed. Fault_Number[0-9] refers to the signals that control which gate in the design is currently in TEST mode. The Dataout[0-32] signals are the signals used to read back the signature of the BIST scheme. Not all of the signals were needed for the designs actually emulated. The signal space was allocated on the basis of future expansion of the system. Each Port on a module can only be used for one set of signals. For instance, the six unused signals on Module 1, Port 3 cannot be used unless the Fault_Number vector size is increased.

Current experimentation leads us to believe that the largest design possible would number in the area of 600 gates. This is based on each fault model gate taking up approximately 2.5 CLBs including all overhead, a 95% utilization of the CLBs and using four XC4010s each containing 400 CLBs. The 95% utilization is not unreasonable, as some single chip designs built in our laboratory have reach a 100% utilization with the 4000 series. However, when dealing with a multi-chip design, 95% utilization is probably all that can be expected. The utilization approximation stated here is a heuristic figure, as the actual CLB utilization is design dependant. The I/O utilization/capacity is a constraining factor, as the I/O needed will most likely increase as the design size increases due to the number of split nets in a partitioned design.

Table 3.3 Listing and Distribution of Control Signals

Signal Name	Module Configuration
Clock_Enable	Module 0, Output 0
Reset	Module 0, Output 1
ALU_S0	Module 0, Output 2
ALU_S1	Module 0, Output 3
CSTP_Mode	Module 0, Output 4
Fault_Control[0-5]	Module 1, Port 0
Fault_Number[0-7]	Module 1, Port 2
Fault_Number[8-9]	Module 1, Port 3
Dataout[0-7]	Module 2, Port 0
Dataout[8-15]	Module 2, Port 1
Dataout[16-23]	Module 2, Port 2
Dataout[24-31]	Module 2, Port3

The system size could be increased if commercial fault grading tools are used to preprocess the design. The preprocessing would isolate a relatively small set of gates with detectabilities below a threshold. These gates would incur the overhead while the remaining gates would be implemented directly.

3.7 Circuits Emulated

The initial circuit designed for the emulator consisted of an eight by eight bit signed multiplier feeding into a sixteen bit ALU. This design was chosen because it approached the maximum capacity of the emulator. The circuit should not have been difficult to test, as both the multiplier and ALU are 100% observable. The circuit can be seen in Figure 3.7. The circuit was emulated using both the LFSR and CSTP BIST schemes.

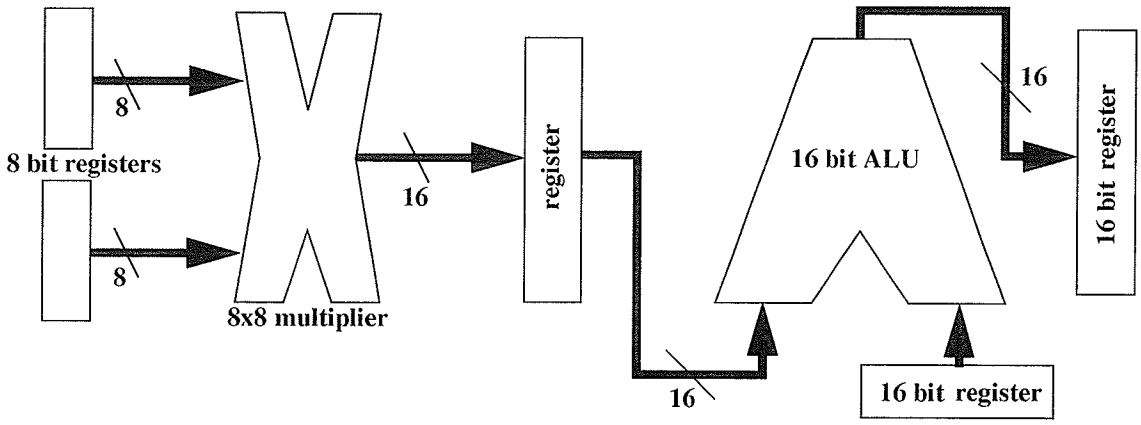

Figure 3.7 ALU - Multiplier Circuit used for Initial Tests

The above design was mapped to 1342 CLBs, which were partitioned to all four of the available XC4010s. A design of this size pushed the design tools to their maximum capabilities, a fact illustrated by the need to get special permission from NeoCAD to acquire a license to partition to more than three devices. When originally placing and routing this design, some problems were encountered. Global buffers associated with some of the control signal were not able to be placed on some of the FPGAs. This turned out to be a software related problem and not a problem relating to the design or the partitioning results. The solution to this was to manually place the global buffers, and lock their placement before running the automatic place and route tools.

The first trial of the emulator was run using the LFSR scheme. This trial was successfully run at a speed of 500 KHz. The timing was generated by the built-in Xilinx oscillator and controlled by the test station. When a timing sequence was complete a trigger signal was generated on the emulator which was read by the tester.

The second trial, using the CSTP scheme was not successfully run. The results generated by this trial were not reproducible. The CSTP scheme had successfully modelled a pseudorandom number generator. Unfortunately this was not the goal of the research. The problem was attributed to poor clock distribution, and too high of a clocking

frequency. Although the maximum operating frequency of each of the FPGAs was known, it was not know how many paths had been split during the partitioning process. If a long path had been split four times, then the maximum operating frequency of the system could be four times the maximum operating frequency of the slowest FPGA. If the path had been split ten times, the operating frequency could be reduced by ten. Because of this unknown factor, it was decided to slow the CSTP scheme down to 125 KHz.

To implement this clock speed, the design had to be re-compiled by the NeoCAD tools. After partitioning of the slower design, a fatal error was produced by the place and route software. The error was attributed to the software. It involved the software being unable to implement the Xilinx wide-edge decoder macros. This error is scheduled to be fixed for the 7.0 release of the NeoCAD tools. Attempts were made to re-partition the design, in hopes that the error would not occur (it becomes manifest because the partitioning results), however, this was not successful.

Due to the problems associated with the partitioning process, it was decided to implement a single FPGA design. A subset of the previous design was chosen. The sixteen bit ALU was now the focus. This design was able to fit into exactly one XC4010, using all 400 of the CLBs. The design was able to be placed and routed without difficulty. This demonstrates the excellent routing resources available for the 4000 series.

Both the LFSR and CSTP schemes were emulated, and the results were as expected. The tests implemented in Chapter two could not be repeated with this emulator, as there is currently no process to evaluate the test signatures after each clock cycle. New methodologies, implemented on the same hardware, could produce similar results, but probably at the expense of other factors.

3.8 Xilinx Emulator- Rants and Raves

The worst characteristic of this implementation, is the large hardware overhead. The hardware overhead increases the delays on the FPGAs due to the large number of global signals distributed throughout the chip. Operating frequencies cannot currently be expected to exceed one to two MHz. The operating frequency could be increased if more care is taken in the design stage. Clock distribution for this emulator did not exploit the dedicated clocking nets. It should be noted that the clock net was not the constraining factor for the speed of the emulator, however, if the clock did use the dedicated routing, one more of the global buffers would have been available.

Another step in the design flow that may have increase the efficiency of the emulator is the creation of macros. Most of the parts used in the emulator, were mapped to the CLBs at the lowest level of the design. If the designer maps specific instances of the design individually, that portion of the design may be fixed in terms of CLB usage. This will make for a more efficient mapping and possibly partitioning, in terms of speed. It will insure that parts of the design that are meant to be closely associated in a space will stay that way. This may not lead to the most efficient use of the hardware resources on the FPGA. By creating hard macros, the mapping software will not be allowed to map into some CLBs that are not fully utilized. Because the CLBs are not fully packed, the emulator gate capacity would decrease from its current estimation.

The HP D20 test software was not the most efficient method of controlling the emulator. The software was available and working, and so was manipulated to fit the emulator application. The VXI test station is an excellent platform for control hardware, and in the near future the HP VEE-TEST software will be available to act as the software interface for the test station. This may allow for the clock signal to be generated and controlled directly from the test station.

3.9 Chapter Summary

In this chapter a novel fault emulation scheme based on an Aptix FPCB, and four Xilinx 4010 FPGAs has been presented. This emulator utilized the presence of FPIDs and FPGAs on the same board to implement multi-chip designs. The emulator relied on the ability to create multi-chip designs in order to allow the extra-hardware overhead needed to compensate for the lack of partial dynamic reprogrammability.

The tests performed showed that is possible to create multi-chip designs. At this time creation of multi-chip designs on a large scale is not a trivial process. For this process to become transparent to the user, many improvements have to be made in both the tools available and the design methodology in general. As many of the tools used here are in their infancy, it is expected that emulators based on such a system will become realizable in the future. As well, if such a process would be commercialized it is expected that the design tools would be fully integrated, with a clear design methodology.

CHAPTER 4

CONCLUSIONS AND FUTURE WORK

This thesis has demonstrated two distinct implementations of an FPGA based emulation system. The rational for designing an emulation system was to showcase the applications of reprogrammable hardware. Most applications using FPGAs do not take advantage of the reprogrammable nature of the device. As these devices have matured, the possible applications of the technology has also grown, and the two implementations presented here, highlight two different features of FPGA technology.

The first emulator, discussed in Chapter 2, highlighted the partial dynamic reconfiguration possible when using the Algotronix CAL1024 FPGA. This feature was exploited when dynamically inserting fault models into the designs. The emulator was also used to demonstrate the efficiency of emulation over simulation. This was demonstrated by emulating a BIST technique that was not well suited for parallel fault simulation. The two BIST techniques examined were LFSR based and CSTP based.

The emulator was also used to evaluate the CSTP scheme. Result from the tests performed on this BIST scheme, revealed that CSTP appears to be a viable low overhead alternative to other traditional BIST schemes.

The second emulator, discussed in Chapter 3, highlighted the scalability of an emulation system using both FPGAs and FPIDs. This system did not use the dynamic reconfiguration process as in the previous chapter. Instead this emulator relied on capacity

and scalability to implement all test overhead, including fault models and control structures. New software was used to implement the partitioning of the design. A methodology for partitioning was examined, and partially implemented. Failures in the current software revealed the inadequacies of this relatively new technology. Improvements made in the future will make such applications much more practical and useful.

4.1 Future Work

Future work on fault emulation systems could produce a commercially viable product. Work on a dynamically reconfigurable emulator, will have to rely on new FPGA architecture, as the Algotronix FPGAs are no longer available.

Continued work on the Xilinx/Aptix based system should include exploring new partitioning strategies in order to increase the speed of the design and to ease the place and route processes. Work should also be done, in examining whether designs that use only two or three of the available FPGAs can be effectively implemented.

Further testing of the CSTP and other non-traditional BIST schemes would be very useful. Determining effects of test-paths, and signature taps could easily be implement on the current emulation board. This work should be carried on by those already in the testing field.

Finally the development of a set of VHDL based "Emulation" libraries would be a significant improvement over the existing schematic capture based design entry. Such a library could be targeted for emulation purposes, and the same design could target a different library for implementation. This would greatly enhance the emulation design flow.

APPENDIX A

TESTING OVERVIEW

A1.1 Introduction

Due to the increasing density of Integrated Circuits (ICs) and ever increasing demands for high product quality in manufacturing and throughout the life-cycle of an IC product, Built-In Self-Test (BIST) has evolved into an integral element of circuit design [10][11]. All BIST techniques require two basic elements: a test pattern generator, and a signature compactor. These two elements serve the purpose of defining the test and storing the results. The test vectors must be able to activate the fault and have the faulty behaviour propagate to a primary output. The signature compaction must be able to capture and retain this faulty behaviour. Both elements are equally important when evaluating the quality of the test.

Various test schemes have used different methodologies to produce these components, ranging from Linear Feed Back Shift Registers (LFSR) to Cellular Automata (CA)[11][12]. These schemes have been widely accepted as standard methods for both test vector generation and signature compaction, and are effective for both. However, both of these schemes have the undesirable element of adding a large hardware overhead to the IC. A proposed alternative to LFSR or CA based schemes is the Circular Self-Test Path (CSTP) scheme. This is a low hardware overhead scheme that is similar to the simultaneous self-test (SST) approach presented in [13].

A1.2 LFSR Based Schemes

LFSRs use modulo 2 based arithmetic to produce polynomial based bit streams that act as the test vector inputs. The LFSR produces a pseudorandom pattern to the inputs of the circuit. The number of the pseudorandom patterns can be determined by the polynomial implemented in the LFSR. For test purposes a maximal length polynomial is chosen to insure the greatest possible range of test vectors. The only vector or state not covered by such a test vector generator is the all zero vector. If a maximal length polynomial is implemented, the initial seed will not effect the test quality to a great degree as long as the initial seed is not the all zero vector.

A maximal length polynomial of degree 4 (as used in an 4 bit LFSR) is

$$\text{Max length 4th order polynomial} = X^4 + X^3 + 1 \tag{A1.2}$$

The implementation of this polynomial can be seen in Figure A1.8 and is derived as follows. The recursive behaviour of a shift register can be described as:

$$a_m = \sum_{i=1}^{n} c_i a_{m-i} \tag{A1.3}$$

where $c_i = 0$ or 1 depending on the feedback. The generating function is stated as:

$$G(x) = \sum_{m=0}^{\infty} a_m x^m \tag{A1.4}$$

where a_m represents the history of the output stage of a shift register where $a_i = 0$ or 1 depending on the state of the output at time t_i. By applying the formula A1.2 into A1.3, G(x) can be simplified to become:

$$G(x) = \frac{c_n}{1 - \sum_{i=1}^{n} c_i x^i} \tag{A1.5}$$

At this point, the denominator

$$f(x) = 1 - \sum_{i=1}^{n} c_i x^i \qquad \text{(A1.6)}$$

will be referred to as the characteristic polynomial of the sequence a_m as well as of the LFSR used to produce it[11].

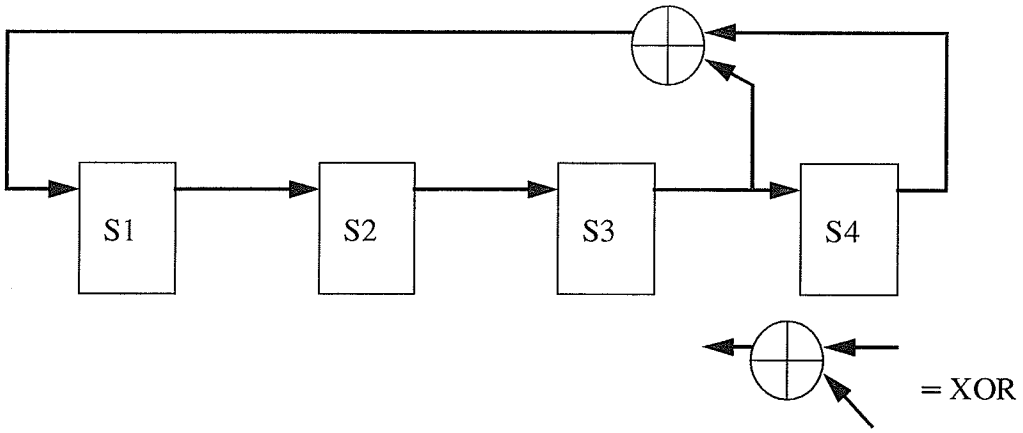

Figure A1.8 *Implementation of a Maximal Length Four Bit LFSR*

From Figure A1.8 it is clear that the recursive definition of the LFSR is

$$a_m = a_{m-3} + a_{m-4}$$

When the above definition is applied to equation A1.5, the polynomial stated in A1.1 is derived. This implementation will produce sequence of 2^4-1 unique four bit outputs.

Signature compaction can take place by using the same LFSR structure as discussed previously. In this case the LFSR performs a recursive compaction. This can occur as a single serial bit stream or in parallel. The end result will be a unique pattern based on the sequence of the previous patterns. In this way the signature compaction acts as a finite state machine, with each state being dependant on the previous one. When one of the

states in the sequence changes, the rest of the states in the sequence will also change. Thus, there can be repeated states for any given set of inputs, however, there should be a unique sequence. This concept is illustrated in Figure A1.9. The first sequence represents the signature compaction occurring with correct outputs from the circuit under test into the

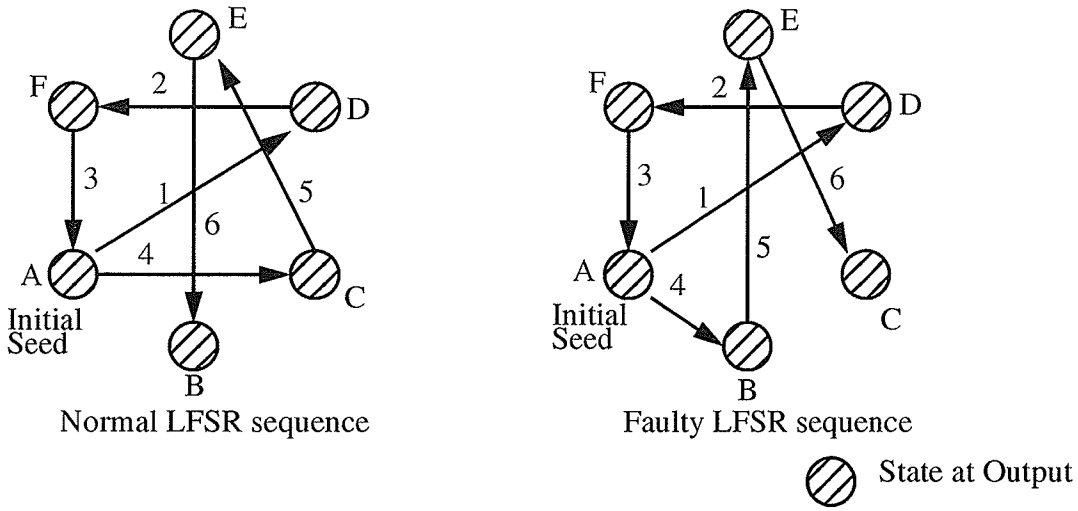

Figure A1.9 *Examples of Two LFSR Signature Compaction Sequences*

LFSR. After six clock cycles the LFSR pattern corresponds to the pattern represented by state B (centre, bottom). In the next sequence, a fault in the circuit has propagated to the output and affects the sequence at the fourth clock cycle. After the third clock cycle the sequence has reached state A, however, instead of changing to state C, as in the normal sequence, the LFSR will go into state B. The sequence then continues through state E and ends on state C. When the user checks the output of the LFSR at the end of the test, it can be seen that the signature of the circuit is not correct. In an actual application there would be more states present. The number of states is equal to 2^n, where n is the number of registers used in the LFSR. Thus for an LFSR of length 16, there would be 2^{16} possible states.

For this kind of signature compaction scheme there is always the possibility of aliasing. Aliasing is defined as the occurrence of a fault free signature after the completion of a faulty sequence. However for a reasonable number of clock cycles the probability of aliasing for such a scheme is very low and is approximated as:

$$P(\text{Aliasing}) \approx \frac{1}{2^n} \tag{A1.7}$$

where n is the length of the LFSR. For an LFSR of length 16 this would result in a probability of aliasing of .00153%.

A1.3 Circular Self-Test Path Based Schemes

Circular Self-Test Path (CSTP) is a testing scheme that uses existing registers in the circuit to both generate the test vectors, and to perform the task of signature compression. This is achieved by augmenting a register with an XOR gate and a multiplexer. Each register's output can be fed to the adjacent register's input where it is XOR'd along with the normal input. This is shown in Figure A1.10. The test path can be extended through all the registers in the circuit, and must include the registers to the primary inputs and primary outputs, as illustrated by the circuit shown in Figure A1.11.

The test vectors are defined as the bit stream to the primary inputs, as is the case with LFSR based testing schemes. The signature compaction takes place throughout the entire test path (as does the test vector generation, the test vectors themselves are defined at the primary inputs). Because of this unique quality, the signature may be tapped from any of the registers in the test path. The quality of the signature compaction may be dependant on the source of the signature. This was not explored in the present research.

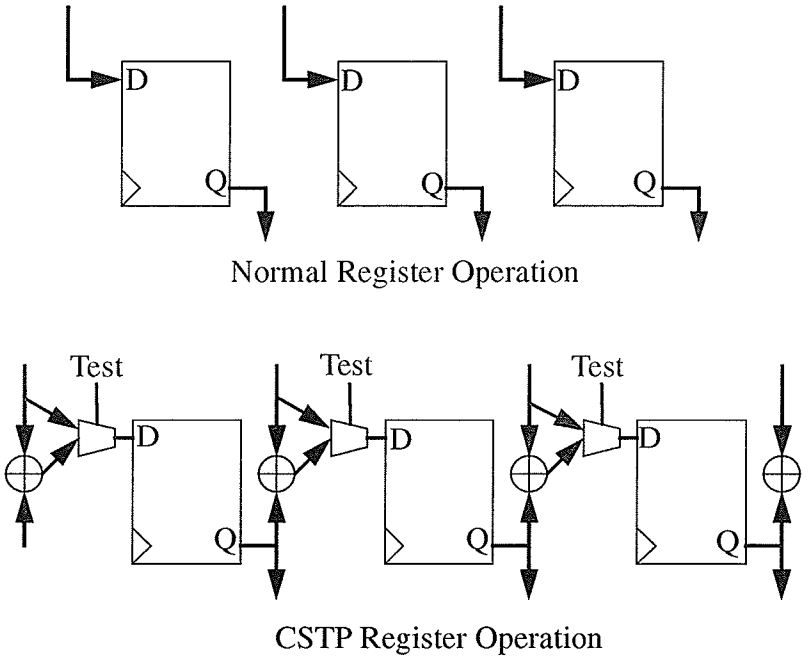

Figure A1.10 *Hardware Required for CSTP scheme*

It is the low hardware overhead, that is to say, the inclusion of a single gate and a multiplexer, that makes the CSTP scheme especially beneficial for BIST. However, due to the feed back loop (the test path) the present state of the inputs to the circuit are dependant on the previous state of the registers in the test path. Because of this relationship, parallel fault simulators can not operate effectively on such a circuit. Normally in parallel fault simulation all the inputs to the circuit are known. Because this is not the case for CSTP, the parallel structure of the simulator cannot be exploited. This means that the simulation time will increase significantly, and a trade-off between hardware overhead and simulation time overhead develops. This trade-off may be one of the reasons that CSTP has not been widely accepted.

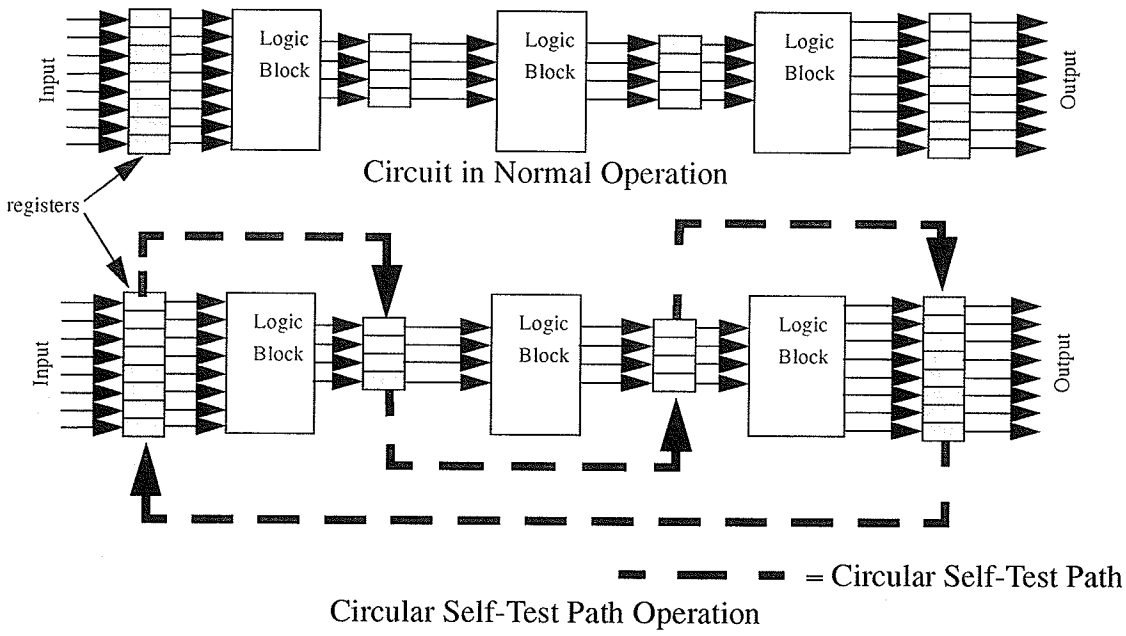

Figure A1.11 *Schematic of Basic CSTP Configuration*

A1.4 Fault Modelling

Every single stuck-at fault has an associated fault model. These fault models physically replace the gate under test during the emulation. As an example, let us examine the two input AND gate. For such a gate there are four possible equivalent fault models. If the output of the gate is stuck at zero, the gate is modelled as a ground. If the output is stuck at one, the gate is modelled as a voltage source. If either input A or B is stuck at one, the associated fault model is a buffer from line B or A respectively. When either input A or B is stuck at zero, the fault model is equivalent to the output of the gate being stuck at zero, which was mentioned earlier. This example is shown in Figure A1.12.

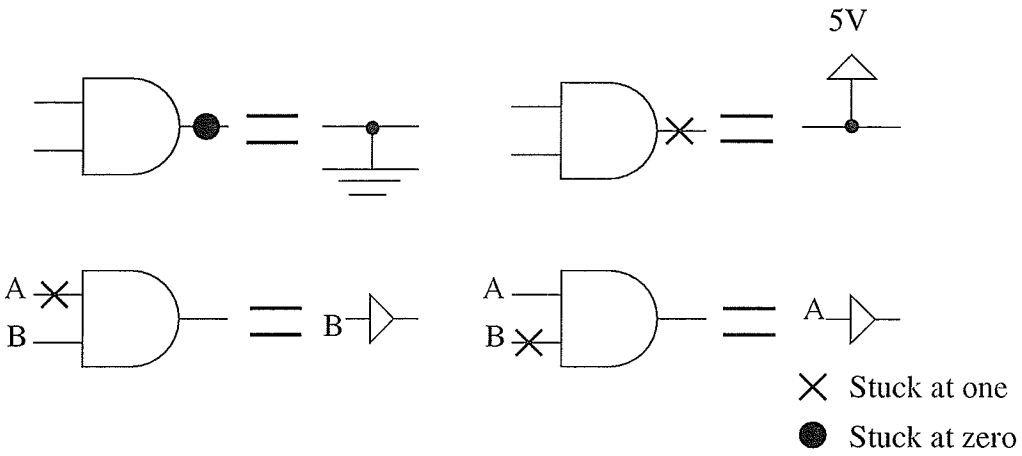

Figure A1.12 *Equivalent Fault Models for a Two Input AND Gate*

Table A1.4 gives a complete listing of two input gates and their associated fault models.

Table A1.4 Fault Models for All Two Input Gates

Gate	Fault	Fault Model
AND	SA1-Output	VDD
	SA0-Output	GND
	SA1-InputA	Buffer-InputB
	SA1-InputB	Buffer-InputA
NAND	SA1-Output	VDD
	SA0-Output	GND
	SA0-InputA	Inverter-InputB
	SA0-InputB	Inverter-InputA
OR	SA1-Output	VDD
	SA0-Output	GND
	SA0-InputA	Buffer-InputB
	SA0-InputB	Buffer-InputA

Table A1.4 Fault Models for All Two Input Gates

Gate	Fault	Fault Model
NOR	SA1-Output	VDD
	SA0-Output	GND
	SA1-InputA	Inverter-InputB
	SA1-InputB	Inverter-InputA
XOR	SA1-Output	VDD
	SA0-Output	GND
	SA1-InputA	Inverter-InputB
	SA0-InputA	Buffer-InputB
	SA1-InputB	Inverter-InputA
	SA0-InputB	Buffer-InputA
XNOR	SA1-Output	VDD
	SA0-Output	GND
	SA1-InputA	Buffer-InputB
	SA0-InputA	Inverter-InputB
	SA1-InputB	Buffer-InputA
	SA0-InputB	Inverter-InputB

GLOSSARY

ASIC - Application Specific Integrated Circuitry

ALU - Arithmetic Logic Unit

BIST - Built-In Self-Test

CA - Cellular Automata

CAL - Configurable Array Logic

CLB - Configurable Logic Block

CSTP - Circular Self-Test Path

EPROM - Erasable Programmable Read Only Memory

EEPROM - Electronically Erasable Programmable Read Only Memory

FPGA - Field Programmable Gate Array

FPID - Field Programmable Interconnect Device

LFSR - Linear Feedback Shift Register

PLD - Programmable Logic Device

RAM - Random Access Memory

ROM - Read Only Memory

SRAM - Static Read Only Memory

BIBLIOGRAPHRY

[1] Albaharn, Cheung, and Clarke, "Area and Tiime Limitations of FPGA based Virtual Hardware" IEEE International Conference on Computing Design, pp 184-189 1994.

[2] Andre DeHon, "DPGA-Coupled Microprocessors: Commodity IC's for the Early 21st Century" IEEE Workshop on FPGAs for Custom Computing Machines, pp. 31-39, 1994.

[3] J. Arnold et al, "The Splash2 Processor and Applications" Proceedings from the International Conference on Computer Design, 1993.

[4] R. G. Shoup, "Programmable Cellualr Logic Arrays", PhD Thesis, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1970.

[5] Rahul Razdan, and Michael Smith, "A High-Performance Microarchitecture with Hardware-Programmable Functional Units" Micro-27, November, 1994.

[6] Stephen D. Brown, Robert J. Francis, Jonathan Rose, and Zvonko G. Vranesic, Field Programmable Gate Arrays, Kluwer Academic Publishers, Boston, 1993.

[7] David Galloway, David Karchmrer, Paul Chow, David Lewis, and Jonathan Rose, "The Transmogrifier: The University of Toronto Field-Programable System", The Canadian Workshop on Field Programmable Devices, 1994.

[8] Pak Chan, Martine Schlag, and Mrcelo Martin, "BORG: a Reconfigurable Proto-

typing Board using Field-Programmable Gate Arrays" ACM/SIGDA Workshop on FPGAs, pp 47-51, 1992.

[9] Michael Thornburg, Steven Cassselman, and John Schewel, Users Guide, EVC1s, Virtual Computer Corporation, 1994.

[10] E. J. McClauskey, "Built-In Self-Test Techniques", IEEE Design and Test of Computers, pp. 21-36, April 1985.

[11] P. H. Bardell, W. H. McAnney, and J. Savir, Built-In test for VLSI: Pseudorandom Techniques, John Wiley &Sons, 1987.

[12] P. D. Hortensius, R. D. McLeod, W. Pries, D. M. Miller, and H. C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 8, No. 8, pp. 842-859, August 1989.

[13] P. H. Bardell and W. H. McAnney, "Self-Testing of Multichip Logic Modules", Proceedings of the International Test Conference, pp. 200-204, 1982.

[14] Frank F. Tsui, LSI/VLSI Testability Design, McGraw-Hill Book Company., New York, 1987.

[15] Algotronix Ltd. Configurable Array Logic User Manual, Edinburgh, UK, 1991.

[16] Texas Instruments Incorporated, The TTL Data Book for Design Engineers, Dallas, Texas, 1981.

[17] J. L. A. Hughes and E. J. McClauskey, "Multiple Stuck-at Fault Coverage of Single Stuck-at Fault test Sets", Proceedings of the International Test Conference, pp. 368-374, 1986.

[18] A. Krasniewski, and S. Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits", IEEE Transactions on Computer-Aided Design, Vol. 8, pp. 425-428, January, 1989.

[19] J. A. Waicukauski, E. Lindbloom, B. R. Rosen and V. S. Igengar, "Transition Fault

Simulation", IEEE Design and Test of Computers, pp. 32-38, April, 1987.

[20] Xilinx, The Programmable Logic Data Book, San Jose, California, 1994.