Analysis of Multiple Scan Path Architecture

A dissertation submitted in partial fulfillment
of the requirements of M.Tech.(Computer Science)
degree of Indian Statistical Institute, Kolkata

by

Ranjan Kumar

under the supervision of

Prof. Bhargab B. Bhattacharya

**Indian Statistical Institute,
Kolkata-700 108.**

July 19, 2004

Indian Statistical Institute,

203, Barrackpore Trunk Road,

Kolkata-700 108.

Certificate of Approval

This is to certify that this thesis titled "**Analysis of Multiple Scan Path Architecture** " submitted by **Ranjan Kumar** towards partial fulfillment of requirements for the degree of M.Tech in Computer Science at Indian Statistical Institute, Kolkata embodies the work done under my supervision.

B. B. Bhattacharya

Prof. Bhargab B. Bhattacharya. 21-7-04
Advance Computing and Microelectronics Unit,
Indian Statistical Institute.
Kolkata-700 108.

Acknowledgements

I take pleasure in thanking Prof. Bhargab B. Bhattacharya for his valuable guidance throughout the dissertation period. His encouraging words have always kept my spirits up.

I express my deep gratitude to my teachers in the course. I would like to thank Faculty and Research Scholars of Advance Computing and Microelectronics Unit, ISI, Kolkata. Finally I take this opportunity to thank my classmates and friends for their help, support and lighter moments throughout the course.

Ranjan Kumar

Ranjan Kumar 22.07.04

Abstract

Scan design has been applied to the problem of testing sequential circuits as a means of increasing the controllability and observability of the circuit. Yet the Scan based testing suffer from prolonged test application time and excessive test power due to numerous shift operations. During the test mode , filling the test data requires shifting the bits one by one into the scan chain ,thus creating the increased switching activities in the flip-flops. Average test power can be minimized by minimizing the switching activities in the flip-flops. Modifying the scan chain judiciously may lower both the test application and average test power by minimizing the switching activities. In recent years number of scan architectures are proposed in order to minimize the switching activities and test application time. We have investigated the power consumption , area overheads and hardware overheads of full scanned version of ISCAS'89 benchmark circuits for different architecture namely Single Serial Scan Architecture and Double Tree Scan architecture.

Contents

List of Figures

List of Tables

0.1 Introduction

Testing of a system is an experiment in which the system is exercised and its response is analyzed to ascertain whether it behaved correctly. If incorrect behavior is detected, a second goal of a testing experiment may be to diagnose, or locate, the cause of the misbehavior.

Testing is a generic term that covers widely different activities and environment, such as:

1. one or more subsystems testing another by sending and receiving messages
2. a processor testing itself by executing a diagnostic program.
3. automatic test equipment (ATE) checking a circuit by applying and observing binary patterns

The stimuli and the response defining a testing experiment correspond to the type of information processed by the system under test. The type of information processed by the system under test depends on the level of abstraction used to define the system. Various levels of abstraction are:

1. logic level: the information processed at this level is represented by discrete logic values. The classical representation uses binary logic values (0 and 1). More accurate models, however, often require more than two logic values
2. register level: the data part of the information consists of words, while the control part is defined in terms of binary values. Since the words are stored in registers, this level is referred to as the register level.
3. instruction set level: at this level both the data part and the control part of the information are organized as words.
4. processor level: at this level of abstraction we can regard a system as processing sequences of instructions, or programs, that operate on blocks of data, referred to as data structures.
5. system level: the system can be viewed as composed of independent systems which communicate via blocks of words called messages.

0.1.1 Errors and Faults

An instance of an incorrect operation of the system being tested or unit-under-test (UUT) is referred to as an error. The concept of error has different meanings at different levels. For example, an error observed at the diagnostic program level may appear as an incorrect result of an arithmetic operation, while for ATE an error

usually means an incorrect binary value. The causes of the observed errors may be design errors, fabrication errors. fabrication defects, and physical failures. Design Errors: Some examples of design errors are:
1 . incomplete or inconsistent specifications of the system.
2 .incorrect mappings between different levels of design.
3. violations of design rules. The space of design errors is not well defined and the set of design errors is not Enumerable. Consequently. it is impossible to develop test generation Algorithms or rigorous quality measures for design verification tests.
Fabrication Errors and Defects: Some examples are:
1. wrong components.
2. incorrect wiring.
3. shorts caused by improper soldering.
4. improper doping profile.
5. mask alignment errors.

Physical Failures

Physical failures occur during the lifetime of a system due to component wear-out and/or environmental factors. For example. aluminum connectors inside an IC package thin out with time and may break due to electron migration or corrosion. Some physical failures, referred to as "infancy failures," appear early after fabrication. Fabrication errors, fabrication defects, and physical failures are collectively referred to as physical faults. According to their stability in time, physical faults can be classified as:
-permanent faults: always being present after their occurrence.
-intermittent faults: existing only during some intervals.
-transient faults: a one-time occurrence caused by a temporary change in some environmental factor.

It is the goal of the testing process to detect these errors and/or faults and ensure that that the end user gets to operate a correct system.

0.1.2 Modeling and Simulation

The way we represent a system has important consequences for the we simulate it to verify its correctness, the way we model faults and simulate it in the presence of faults. and the way we generate tests for it.
We can model a system in following ways:

-functional model: A functional model of a system is a representation of its logic function. The logic function is the I/O mapping that deals only with value transformation. At any level of abstraction. this model represents a system as a black box with its I/O specification.

-behavioral model: A behavioral model consists of a functional model coupled with the representation of the associated timing relations between the I/O signals.

-structural model: A structural model is often hierarchical representation of a system in terms of interconnection of smaller components. The bottom level components are called primitive elements. The functional or behavioral model of primitive elements is known and denoted by its type. A structural model always carries (implicitly or explicitly) information regarding the function of its components.

-external model: an external model is the model viewed by the user. An external model can be graphic (e.g., a schematic diagram) or text based (using Hardware Description Language).

-internal model: An internal model of a system is represented by data structures and/or programs inside a computer.

Logic simulation is a testing experiment that uses a model of a designed system. The model can be exercised by stimulating with a representation of the input signals. Logic simulation determines the evolution in time of the signals in response to an applied input signal. It is also called design verification testing. It is used for detecting errors during design.

Fault simulation consists of simulating a circuit in the presence of faults. Comparing the fault simulation results with those of the fault-free simulation of the same circuit simulated with the same applied test T, we can determine the faults detected by T.

0.1.3 Test Generation

Test generation is the process of determining the stimuli necessary to test a digital system. Test generation can be fault oriented or function oriented. In fault oriented test generation, one tries to generate tests that will detect (and possibly locate) specific faults. In function oriented test generation, one tries to generate a test that, if it passes, shows that the system performs its specified function. Test generation depends primarily on the testing method employed.

0.1.4 Test Evaluation

Test evaluation refers to determining the effectiveness, or quality, of a test. The quality of a test is measured by the ratio of the number of faults it detects and the total number of faults in the assumed fault universe: this ratio is referred to as the fault coverage. Test evaluation is usually done in the context of a fault

0.1.5 Diagnosis and Repair

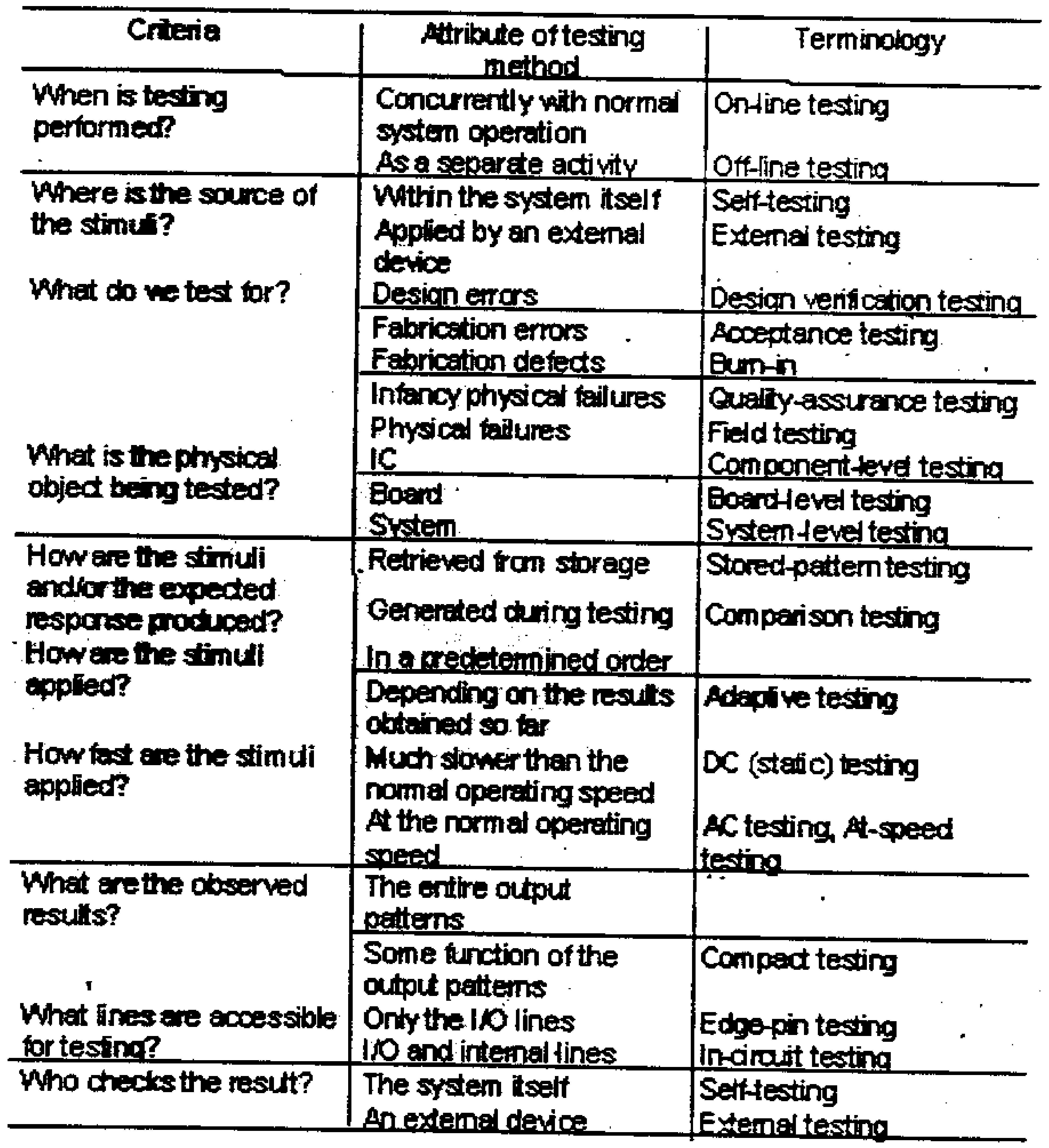
Two types of approaches are available for fault diagnosis:

Cause-effect analysis: This approach enumerates all the possible faults (causes) existing in a fault model and determines, before the testing experiment, all their corresponding responses (effects) to a given applied test. This process, which relies on fault simulation, builds a database called a fault dictionary. The diagnosis is a dictionary look-up process, in which we try to match the actual response of the unit under test with one of the precomputed responses. If the match is successful, the fault dictionary indicates the possible faults in the unit under test.

Effect-cause analysis: An effect-cause analysis processes the actual response of the unit under test and tries to determine directly only the faults (cause) that could produce that response.

0.1.6 Types of Testing

Testing methods can be classified according to many criteria as summarized by the following table:



Criteria	Attribute of testing method	Terminology
When is testing performed?	Concurrently with normal system operation	On-line testing
	As a separate activity	Off-line testing
Where is the source of the stimuli?	Within the system itself	Self-testing
	Applied by an external device	External testing
What do we test for?	Design errors	Design verification testing
	Fabrication errors	Acceptance testing
	Fabrication defects	Burn-in
	Infancy physical failures	Quality-assurance testing
	Physical failures	Field testing
What is the physical object being tested?	IC	Component-level testing
	Board	Board-level testing
	System	System-level testing
How are the stimuli and/or the expected response produced?	Retrieved from storage	Stored-pattern testing
	Generated during testing	Comparison testing
How are the stimuli applied?	In a predetermined order	
	Depending on the results obtained so far	Adaptive testing
How fast are the stimuli applied?	Much slower than the normal operating speed	DC (static) testing
	At the normal operating speed	AC testing, At-speed testing
What are the observed results?	The entire output patterns	
	Some function of the output patterns	Compact testing
What lines are accessible for testing?	Only the I/O lines	Edge-pin testing
	I/O and internal lines	In-circuit testing
Who checks the result?	The system itself	Self-testing
	An external device	External testing

Figure 1: Types of Testing

0.2 Design for Testability

Design for testability (DFT) techniques are been used to reduce the cost of testing by introducing testability criteria early in the design stage. The testability of a circuit is an abstract concept that deals with a variety of costs associated with testing. By increasing the testability of a circuit, it is implied that some function of these costs is being reduced, though not necessarily each individual cost .

Design for testability techniques are design efforts specifically employed to ensure that a device is testable. The following questions characterize testing of complex systems:

-can tests that detect all faults be assured?

-can test development time be kept low enough to be economical?

-can test execution time be kept low enough to be economical? Design for testability (DFT) refers to those design practices that answer the above questions in the affirmative.

Cost of testing and Test complexity

Test complexity can be converted into costs associated with a testing process. There are several facets to this cost, such as the cost of test pattern generation, the cost of fault simulation and generation of fault location information, the cost of test equipment and the cost related to the testing process itself. Because these costs may be high and may even exceed design costs, it is important that they be kept within reasonable bounds. One way to accomplish this goal is by the process of DFT.

Testability of a Circuit (Controllability and Observability)

The testability of a circuit is an abstract concept that deals with a variety of costs associated with testing. By increasing the testability of a circuit, it is implied that some function of these costs is being reduced. Two important attributes related to testability are controllability and observability. Controllability of a circuit is the ability to establish a specific signal value at each node in a circuit by setting values on the circuit's inputs. Observability of a circuit is the ability to determine the signal value at any node in a circuit by controlling the circuit's inputs and observing its outputs. Many of the DFT techniques are attempts to increase the observability or controllability of a circuit design. The idea is to modify the design of a circuit to enhance its controllability and observability values. Most of the DFT techniques try to achieve this goal by either the resynthesis of an existing design or the addition of extra hardware to the design. Most approaches require circuit modifications and affect such factors as area, I/O pins, and circuit delay. The values of these attributes usually increase when DFT techniques are employed. Hence, a critical balance exists between the amount of DFT to use and the gain achieved.

0.2.1 Testability measure programs

A straightforward method for determining the testability of a circuit is to use an automatic test pattern generation (ATPG) program to generaté the tests and the fault coverage. The running time of the program, the number of test patterns generated, and the fault coverage provide a measure of the testability of the circuit. The difficulty with this approach is that there is large expense involved in running the ATPG program and also the ATPG program may not provide sufficient information on how to improve the testability of a circuit with poor testability. A number of programs have been written to estimate the testability of a design without actually running an ATPG program.

Some examples are: TMEAS, SCOAP, TESTSCREEN, CAMELOT, and VICTOR.

These testability measure programs implement algorithms that attempt to predict, for a specific circuit, the cost (running time) of generating test patterns. In the process of calculating the testability measure, information is developed identifying those portions of the circuit that are difficult to test. This information can be used as a guide to circuit modifications that improve testability.

The technique used to demonstrate that a given testability measure program does indeed give an indication of circuit testability is to run both the testability measure program and also an ATPG program on a number of different circuits. A monolithic relation between the run times of the two programs is offered as proof that the testability measure program produces a good estimate of circuit testability.

All the testability measure programs base their estimates of testability on controllability and observability values for each circuit component. They differ in the precise definitions used for obtaining these estimates.

0.3 Scan Design

The most popular structured DFT technique used for external testing is referred to as scan design since it employs a scan register. There are several forms of scan designs: they differ primarily in how the scan cells are designed.

The basic idea of Scan Design method is to obtain controllability and observability for flip-flops. This is done by adding a test mode to the circuit such that when the circuit is in this mode, all flip-flops functionally form one or more shift registers. The inputs and outputs of these shift registers are made into primary inputs and primary outputs. Thus, using the test mode, all the flip-flops can be set to any desired states by shifting those logic states into the shift register. Similarly, the states of the flip-flops are observed by shifting the contents of these scan registers out.

One important advantage of Scan Design technique is that for full serial integrated scan design or for isolated serial scan design where the scan register can both control and observe all the storage cells of a sequential circuit, the test generation problem transforms from a sequential circuit problem to a combinational circuit problem.

0.3.1 Scan Design Rules

In order to be able to make a circuit scan-testable, the designer must adhere to certain rules during functional design. In general, these rules depend upon the specific design environment. The following four rules are found to be useful:
- only D-type flip-flops should be used.
- At least one primary input pin must be available for test.
- All flip-flop clocks must be controllable from primary inputs.
- Clocks must not feed data inputs of flip-flops.

0.3.2 Scan Design Testing

Testing of scan design is done in two phases
1. Shift test: Testing of scan registers is done by shift test. The circuit is first set in test mode by setting the Test Control(TC) signal to 0. a toggle sequence 00110011 is applied through the shift using normal clock signal. This sequence produces all four transitions- 0 -¿ 0, 0 -¿ 1, 1 -¿ 0, 1 -¿ 0 - in each flip-flop and the output is observed.

2. Second phase: A combinational ATPG program is used to generate test vectors assuming all flip-flop outputs are completely observable and all flip-flop inputs are completely controllable. The test generation problem reduces to combinational circuit problem. After setting the flip-flops, the state input vector is applied and the output is observed. The process is repeated as shown in the following figure:

0.3.3 Parameters of scan design

The following are important parameters for any scan design:
1. Test application time : Test application time usually depends on the number of scan flip-flops between SCANCIN and SCANOUT terminal. If n is the number of scan flip-flops between SCANIN and SCANOUT and V is the total number of test vectors, then
Scan Test Length = (V+1) x n + V
If the testing of scan registers is also included, the scan test can be written as :
Scan Test Length = n + 4 + (V+1) x n - V
2. Hardware overhead : Hardware overhead increases the area overhead as well. The actual area overhead depends on layout details of the circuit. The hardware overhead can be expressed in terms of gate overhead. Each flip-flop adds an overhead of four gates. The total gate overhead is thus computed as follows

Gate Overhead of scan = $\frac{4 \times n \times 100\%}{n_g}$

3. Performance overhead : Scan design leads to performance degradation due to delay caused by multiplexors attached with flip-flops. In general, scan design can reduce the clock speed by 5 to 10
4. Power consumption : The power consumption depends mainly on the switching activity. Switching is the transition either from 0 -¿ 1 or from 1 -¿ 0 in scan flip-flops.

0.4 Multiple Scan Path Architecture

0.4.1 Multiple Scan Registers

To reduce the time of scan test, flip-flops can be arranged as multiple scan registers. Multiple scan chain architecture enables the possibility of distributing scan flip-flops among any number of shift registers, each having a separate SCANIN and SCANOUT pins. If extra pins are not available, added fanouts from normal primary input pins can provide SCANIN to scan chains. This is possible because the normal primary inputs and SCANIN are never simultaneously used. Similarly, the SCANOUT signals can be multiplexed with the normal primary output pins under the control of the test control (TC) signal. Just one test control (TC) pin essential. The test sequence length is determined by the longest scan shift register.

0.4.2 Parallel Scan Chains

The basic idea of parallel scan is to link all scan chain inputs together using a single scan-in pin and compress the response outputs so that only one scan-out pin is required. With this configuration the test patterns for all the scan chains are same. At the beginning of an ATPG process, random patterns can be applied to the two circuits until a specified fault coverage, say 85them. Since most faults have been detected by the random patterns, each of the remaining test patterns generated by deterministic ATPG may be needed only for a small number of faults.

There are several hardware configurations for parallel scan chains The following figure gives a general configuration Under this configuration all circuits under test

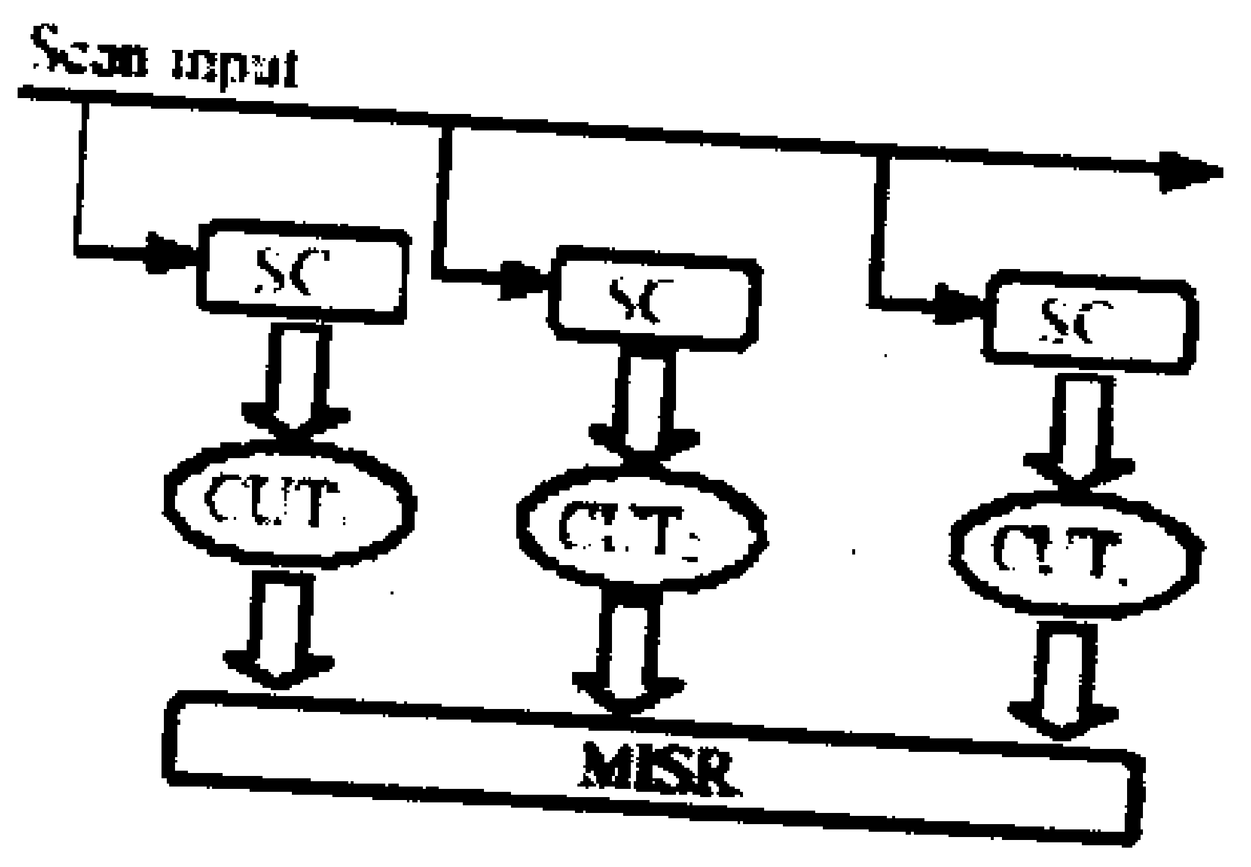


Figure 2: Parallel Scan

receive the same test patterns through the scan input. The patterns are shifted in, then applied to each circuit and the results are compressed by a multiple input signature register (MISR).

0.5 Experiments and Results

Experiments were carried out on the ISCAS89 benchmark circuits for analyzing the parallel scan architecture. The flow diagram as shown below was followed in conducting the experiments.

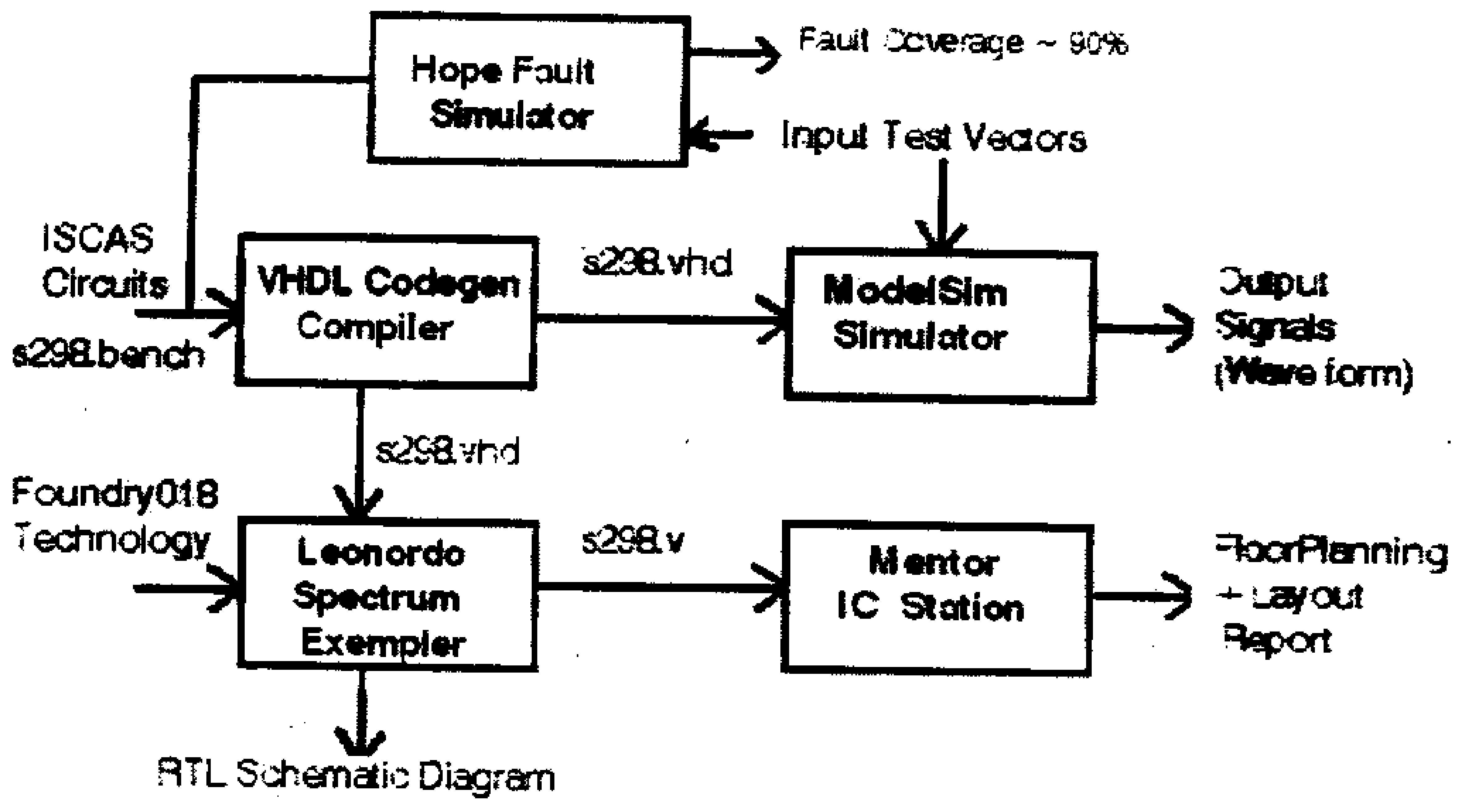


Figure 3: ISCAS'89 Circuit Experimental Flow

The following tables give the results obtained by running the VHDLCodeGen tool on s5378 benchmark circuit :

Result 1 : The results were obtained for various values of multiplicity m. Where m = # of parallel scan paths.

Circuit : s5378 # of scan FFs: 179

Table 1: **Area and I/O Analysis**

Multiplicity, m	Total connections(pins minus nets)	Total internal areamic^2
3	30454	609177.0
5	30456	609167.0
15	30466	609158.0
20	30471	609100.0
30	30481	609129.4

Result 2 : The following results were derived by running the VHDLCodeGen tool on several benchmark circuits using no scan and multiple scan architectures using multiplicity value 3 :

Table 2: **Area Overhead Comparison between No-Scan and Multiple Scan**

Circuit	# of flip-flops	No scan, mic^2	Parallel scan, mic^2	Area overhead,%
s298.bench	14	19792.1	21994.9	11.1
s382.bench	21	28360.0	29664.8	4.6
s510.bench	6	33931.2	34304.0	1.1
s713.bench	19	63939.1	65194.1	2.0
s820.bench	5	50163.4	50486.4	0.64
s1494.bench	6	125167.6	125610.4	0.35

0.5.1 VHDLCodeGen Tool

The VHDLCodeGen tool is designed and developed to generate the VHDL code by incorporating the different scan architectures that we have discussed so far.

The tool is designed to take the input as a standard benchmark circuits ISCAS'89 benchmarks and generate the VHDL code according to the selected architecture. The compiler designed in the tool takes the standard benchmark circuits as an input, the grammar is in the format of out = structure (inp1, inp2, ...). . The structural VHDL code is generated and modifies the scan path according to the selected scan architecture. For instance in the multiple scan architecture the D flip-flop is modified into scan D Flip-flop by adding the extra multiplexors and serially connected in parallel. The VHDLCodegen also generates vhdl code for the clock controller and shift controller required by the double tree scan. architecture.
It implements the following architecture

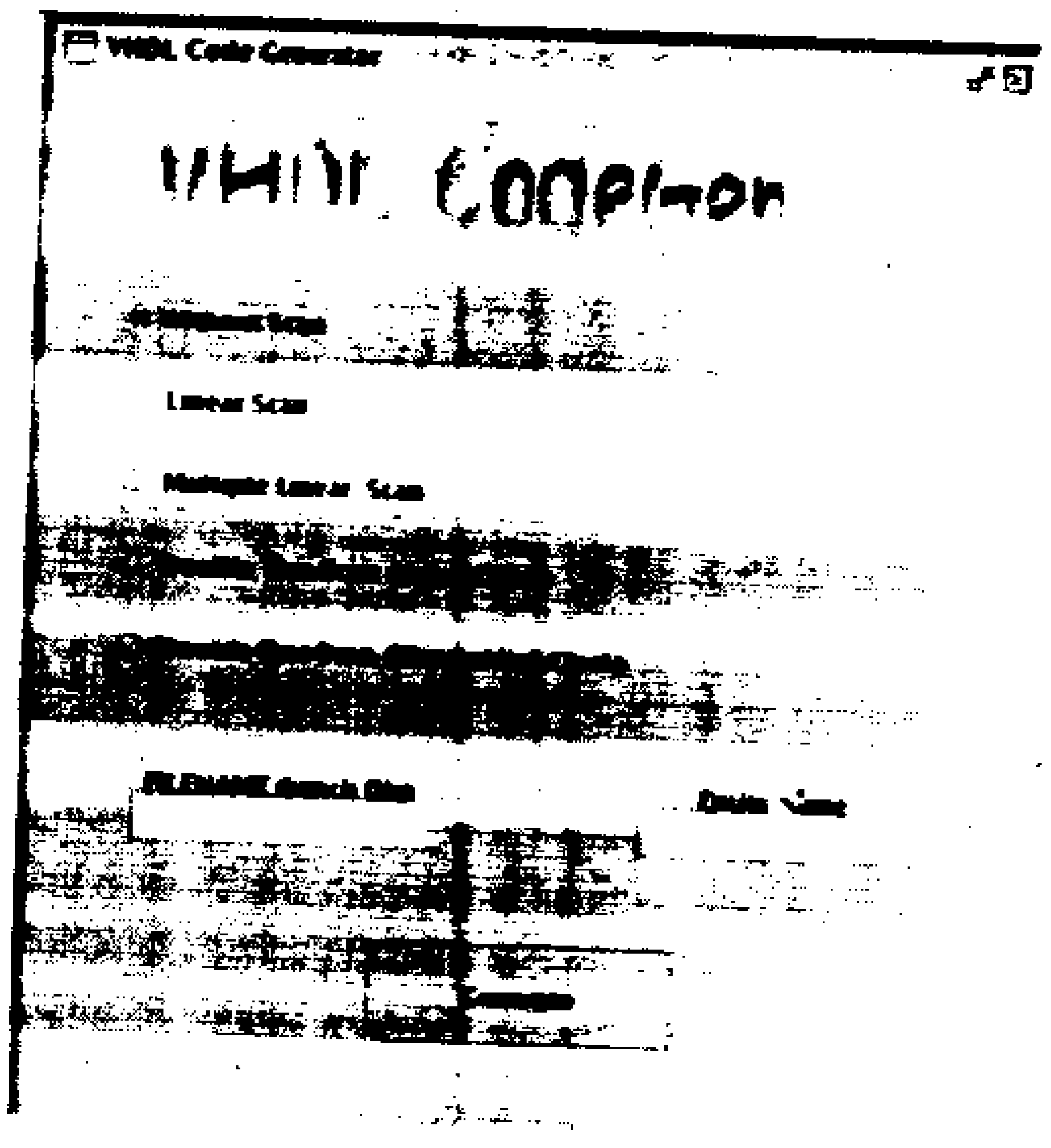


Figure 4: VHDL CodeGen GUI

1. Single Chain Serial Scan Architecture
2. Multiple Chain Serial Scan Architecture
3. Double Tree Scan Architecture with naive clock controller
4. Double Tree Scan Architecture with hierarchical clock controller.

The reports and diagrams enclosed are as follows.

1. RTL Schematic diagram for s298.bench (MS)
2. Hope Fault Simulation report for s344.bench
3. Floor Planning Report for s298.bench(MS)
4. Final Layout diagram of s298.bench(MS)

0.5.2 RTL Schematic diagram for s298.bench (MS)

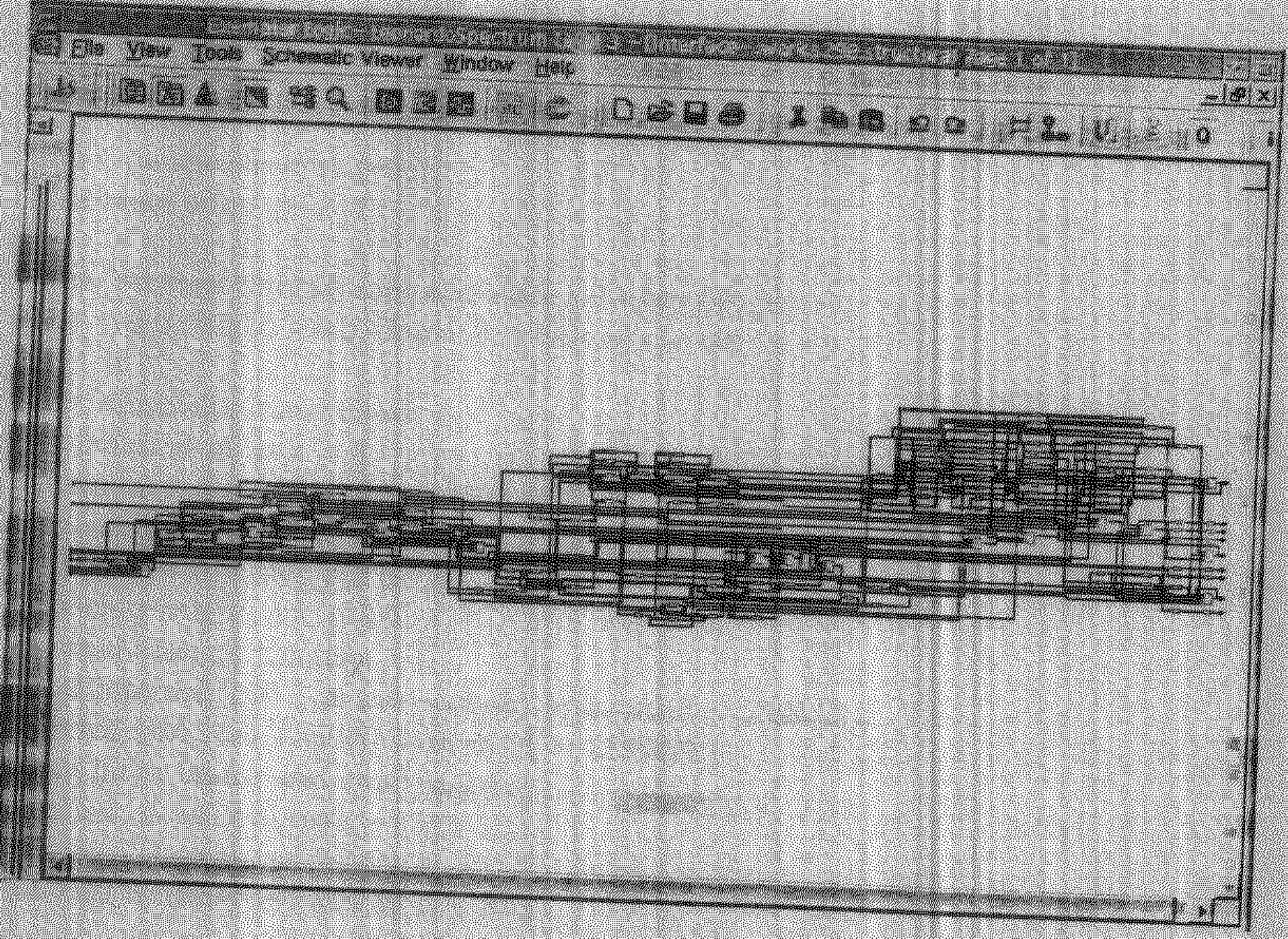


Figure 5: RTL Schematic of s298 circuit

0.5.3 Floor Planning Report for s298.bench(MS)

```
Track spacing:

1 X-track = 660 grids = 1 microns.
1 Y-track = 660 grids = 1 microns.

Site-type-specific data (mic):
Maximum site type = 1

Gaps between internal zone and external rows (mic):
Left = 0.0, Right = 0.0, Top = 0.0, Bottom = 0.0

Row/dimension constraints:
Minimum # of rows = 0
Minimum x dimension, mic = 15.2
Minimum y dimension, mic = 0.0
Max chip width, mic = 3.0
Max chip height, mic = 3.0
Internal zone parameters:
Total internal (cell + macro + chan) area, mic^2 = 21994.9
Internal (std. cell + chan) area, mic^2 = 21994.9
Total large macro area, mic^2 = 0.0
Number of internal rows = 15

Internal routing/cell area ratio = 81

Avg. instance height, mic = 5.8
Avg. channel height, mic = 4.6
Avg. row height rounded to tracks, mic = 10.5
Total connections (pins - nets) = 1655
Internal zone area, mic^2 = 23175.7
Internal zone width, mic = 145.9
Internal zone height, mic = 158.9
Aspect ratio = 0.92
Specified lower aspect ratio = 0.5
Specified upper aspect ratio = 1.5
Overall chip dimensions:
Chip X-dimension, mic: 151.1
Chip Y-dimension, mic: 164.3
```

0.5.4 Hope Fault Simulation Report for s344.scan

```
********************************************************
*                                                      *
*             Welcome to HOPE (version 2.0)            *
*                                                      *
*                Dong S. Ha (ha@vt.edu)                *
*             Web: http://www.ee.vt.edu/ha             *
*  Virginia Polytechnic Institute & State University   *
*                                                      *
********************************************************

******   SUMMARY OF SIMULATION RESULTS   ******

1. Circuit structure
   Name of circuit                          : 9_inputs
   Number of primary inputs                 : 9
   Number of primary outputs                : 11
   Number of flip-flops                     : 15
   Number of gates                          : 160
   Level of the circuit                     : 20

2. Simulator input parameters
   Simulation mode                          : file (inp.tv)

3. Simulation results
   Number of test patterns applied          : 500
   Fault coverage                           : 90.643 %
   Number of collapsed faults               : 342
   Number of detected faults                : 310
   Number of undetected faults              : 32

4. Memory used                              : 134758 Kbytes

5. CPU time
   Initialization                           : 0.017 secs
   Fault simulation                         : 0.200 secs
   Total                                    : 0.217 secs
```

0.6 Conclusion

In the dissertation I have analysed the area overhead between no scan and scan circuit using several benchmark circuits. In the process, VHDLCodeGen tool was developed, which was used to generate the VHDL code for several circuits used in analysis.

Bibliography

[Ash] Peter J. Ashenden. *High level vlsi synthesis.*

[BA] Michael L. Bushnell and Vishwani D. Agarwal, *Essential of electronic testing for digital, memory and mixed signal vlsi circuits.*

[BBBZ03] S. C. Seth B. B. Bhattacharya and S. Zhang, *Double-tree scan: A novel low-power scan-path architecture.* Proceedings 2003 International Test Conference (2003), 470–479.

[CA90] Kwang-Ting Chen and Vishwani D. Agrawal, *A partial scan method for sequential circuits with feedback,* Transactions on Computers **Vol 39 No. 3** (1990), 544–548.

[CW91] R. Camposano and Wayne Wolf, *High level vlsi synthesis,* Kluwer Academic Publishers, 1991.

[DGZ00] A. Paschalis M. Psarakis D. Gizopoulos. N. Kranitis and Y. Zorian, *Low power/energy bist scheme for datapaths,* TProc. VTS (2000), 23–28.

[FBK89] D. Bryan F. Brglez and K. Kozminski, *Combinational pro-files of sequential benchmark circuits,* ISCAS **vol. 14 n. 2** (May 1989), 19291934.

[F.N95] F.N.Najm, *A survey of power estimation techniques in vlsi circuits,* IEEE Transactions VLSI Systems, **vol. 2 no. 4** (January 1995), 446–455.

[IEE87] 1076 Standard IEEE, *Ieee standard vhdl language reference manual,* IEEE Press, 1987.

[MA90] A. D. Friedman M. Abramovici, M. A. Breuer, *Digital systems testing and testable design,* IEEE Press, 1990.

[McC] E. J. McCluskey, *Logic design principles.*

[PC91] Scott R. Powell and Paul M. Chau, *A model for estimating power dissipation in a class of dsp vlsi chips.* IEEE Transactions on Circuits and Systems Vol. 38 No. 6 (June 1991).