CMPE646 VLSI DESIGN VERIFICATION AND TESTING

FINAL PROJECT

On

SEQUENTIAL CIRCUIT TEST PATTERN GENERATION AND FAULT SIMULATION

Submitted by

VEMULAPLLI SRI SAMADARSINI

(JV29859)

jv29859@umbc.edu

LAKSHMI PRIYA PALAPARTHY

(AT97570)

Lakshmp1@umbc.edu

Submitted to

Dr. Naghmeh Karimi

University of Maryland Baltimore County

FALL 2023

INDEX

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Synopsys TetraMax is an ATPG tool used to generate test patterns and to perform fault simulation for the developed VLSI circuits. In this project TetraMax is used to perform Test generation, and Fault simulation for sequential circuit without scan chain, with scan chain, with multiple scan chains and with partial scan.

- **Test Generation**:

Test generation is defined as generating the input test patterns which detects all stuck at faults in the circuit.

- **Fault Simulation:**

Fault simulation is defined as to evaluate the fault coverage obtained by the set of test patterns that are generated by test generation technique.

Generally, with tool or without tool the sequential circuits test pattern generation is more complex than the combinational circuits test pattern generation. So, in order to reduce this complexity, we have some methods called Full Scan, Multiple Scan, Partial scan. This all methods involves the Scan Flip flops (SFF) which indicates the normal D flip flops in the sequential circuits.

- **Full Scan:**

  In full scan design the flip flops of the sequential circuit is replaced as the scan flip flops. The scan design obtains the controllability and observability for flipflops by adding a test mode to the circuit.

- **Multiple Scan Chain:**

  In multiple scan chain technique, the scan flip flops are distributed into the number of scan chains. These SFF are divided among any number of shift registers each having a separate scan in (SI) and scan out (SO) pins.

- **Partial Scan:**

  In partial scan technique, the subset of scan flip flops is considered but not all the flip flops as scanned ones in the given sequential circuit. In partial scan there is a scan and non-scan flip flops which are controlled from separate clock.

In this project Firstly, we performed the deterministic test pattern and random test pattern simulation for the given “circuit.v”. Later all the DFF are replaced by SDFF to build a single scan chain which is the Full scan technique. Thereafter, all those SDFF are divided into multiple scan chains. Moving on considering only some number of flip flops as scan flip flops which is a partial scan. Therefore, we have covered all the above techniques for testing a given sequential circuit using the ATPG tool.

Description of Fault classes

- **Detected (DT):** are the faults identified as detected as hardly by getting the differential output from the golden circuit and faulty circuit.

- **Possibly Detected (PD):** are the faults which output is from the simulation is “X” instead of 1 or 0. So it is hard to decide it is hardly detected.
- **Undetectable (UD):** are the faults which cannot be detected in any condition.
- **ATPG Untestable (AU)**: are the faults which is not detected hardly under the ATPS conditions. Also not considered as redundant faults.
- **Not Detected (ND):** are the faults that test generation tool was not able to create a pattern that can control or observe the faults.

SEQUENTIAL CIRCUIT (NO SCAN CHAIN)

2.1.a) Deterministic Test Pattern Generation

Here using the Tetramax tool, we performed sequential test generation (no scan chain is involved) for given Verilog netlist file named "circuit.v" and generated the set of test patterns which detects all detectable faults in a given circuit. Here, to perform these operations we used the tetra max tool in shell by using the command "tmax -shell". By launching the tetra max in shell mode, we run the "Test_Generation1a.tcl" which contains all the commands that requires to perform the required operations like creating a log file, reading netlists, running DRC, adding faults, and to generate patterns to a file.

After the ATPG is performed, there created the files called "test_gen1a.log", "ATPG_pattern1a.pattern" files in the attached zip file (under part1-1a) which gives us the details of the faults and the patterns generated. From the output files, we observed the total faults, faults statistics, and number of patterns, Test coverage, and time taken by the tool as shown in below Table 2.1.1 and Table 2.1.2.

Table 2.1.1: Faults class statistics of Deterministic test pattern generation for no scan chain

Parameters		**Value**
Total Faults		51372
Fault classes	Detected (DT)	24014
	Possibly Detected (PT)	757
	Undetectable (UD)	1169
	ATPG Untestable (AU)	13261
	Not Detected (ND)	12171

Table 2.1.2: Parameters of Deterministic test pattern generation for no scan chain

Parameters	**Value**
Total Faults	51372
Number of patterns generated	197
Test Coverage	48.59%
Fault Coverage	46.74%
Time taken by CPU	78468.64s (21:47 hrs)

In each case the fault coverage is detected by dividing the detected faults with the total faults.

From the above tables, we represented the observed data in the form of graphs as shown below. Fig 2.1.1 represents the comparison between the test coverage and fault coverage for no scan chain sequential circuit. Fig 2.1.2 represents the fault class statistics for no scan chain sequential circuit.

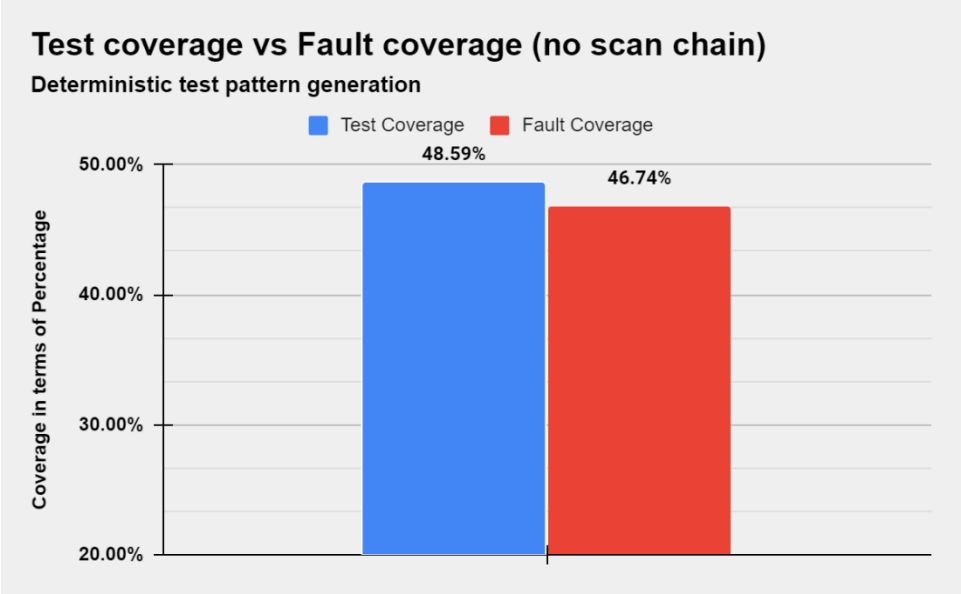

Fig 2.1.1: Graphical representation of Test coverage vs Fault coverage for no scan chain

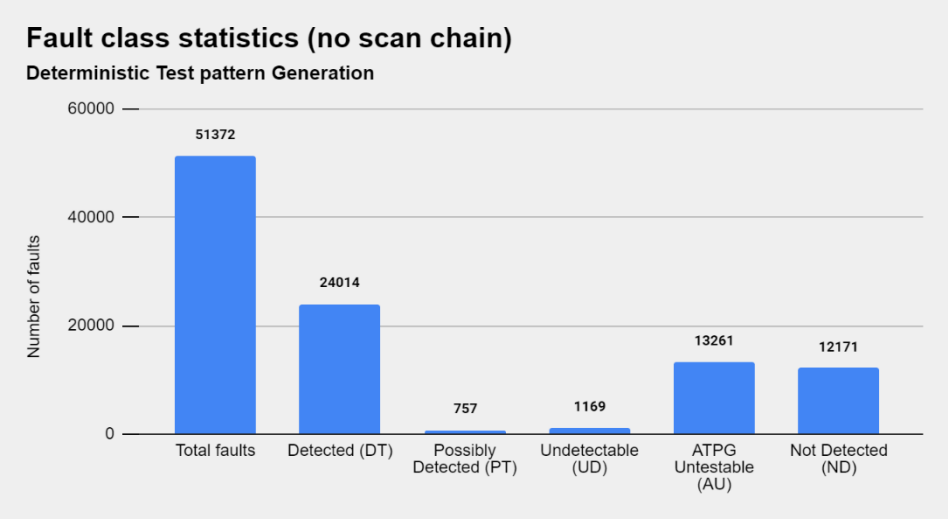

Fig 2.1.2: Graphical representation of Fault classes statistics for no scan chain

2.1.b) Random Test Pattern Simulation

Here, we generate random patterns i.e., 30% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 197 test patterns. So, 30% of those patterns is 60. Hence, we created the 60 random patterns by writing a python code. These generated random patterns are now used to generated '.pattern' file named "random_pattern_30". The '.tcl' file named "Test_Generation_fs_30" contains the required commands to run the input pattern file and generate the "ATPG_pattern_fs_30". Now again we

written another '.tcl' file named "Test_Generation_fl_30" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_30" and the faults left file is generated with the name of "faults_left_fl_30".

Table 2.1.3: Parameters of 30% Random Test Patterns Simulation for no scan chain

Parameters		**30% patterns**	**50% patterns**	**80% patterns**
		Value	**Value**	**Value**
Total Faults		51372	51372	51372
Fault classes	Detected (DT)	5212	5205	5478
	Possibly Detected (PT)	209	177	184
	Undetectable (UD)	664	664	664
	ATPG Untestable (AU)	40713	40713	40713
	Not Detected (ND)	4574	4613	4333
Number of patterns generated		60	99	158
Test Coverage		10.48%	10.44%	10.98%
Fault Coverage		10.14%	10.13%	10.66%
Time taken by CPU		0.00s	0.00s	0.00s

2.1.c) Random Test Pattern Simulation

- Here, we generate random patterns i.e., 50% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 197 test patterns. So, 50% of those patterns is 99. Hence, we created the 99 random patterns by writing a python code. These generated random patterns are now used to generated '.pattern' file named "random_pattern_50". The '.tcl' file named "Test_Generation_fs_50" contains the required commands to run the input pattern file and generate the "ATPG_pattern_fs_50". Now again we written another '.tcl' file named "Test_Generation_fl_50" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_50" and the faults left file is generated with the name of "faults_left_fl_50".

Table 2.1.4: Parameters of 50% Random Test Patterns Simulation for no scan chain

Parameters		**50% patterns**
		Value
Total Faults		51372
Fault classes	Detected (DT)	5205
	Possibly Detected (PT)	177
	Undetectable (UD)	664
	ATPG Untestable (AU)	40713
	Not Detected (ND)	4613
Number of patterns generated		99
Test Coverage		10.44%
Fault Coverage		10.13%
Time taken by CPU		0.00s

- Again, we generate random patterns i.e., 80% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 197 test patterns. So, 80% of those patterns is 158. Hence, we created the 197 random patterns by writing a python code. These generated random patterns are now used to generated '.pattern' file named "random_pattern_80". The '.tcl' file named "Test_Generation_fs_80" contains the required commands to run the input pattern file and generate the "ATPG_pattern_fs_80". Now again we written another '.tcl' file named "Test_Generation_fl_80" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_80" and the faults left file is generated with the name of "faults_left_fl_80".

Table 2.1.5: Parameters of 80% Random Test Patterns Simulation for no scan chain

Parameters		80% patterns Value
Total Faults		51372
Fault classes	Detected (DT)	5478
	Possibly Detected (PT)	184
	Undetectable (UD)	664
	ATPG Untestable (AU)	40713
	Not Detected (ND)	4333
Number of patterns generated		158
Test Coverage		10.98%
Fault Coverage		10.66%
Time taken by CPU		0.00s

From the above tables, we represented the observed data in the form of graphs as shown below. Fig 2.1.3 represents the comparison between the test coverage and fault coverage for 30%, 50%,80% test patterns in no scan chain sequential circuit. Fig 2.1.4 represents the fault class statistics for 30%, 50%,80% test patterns in no scan chain sequential circuit.

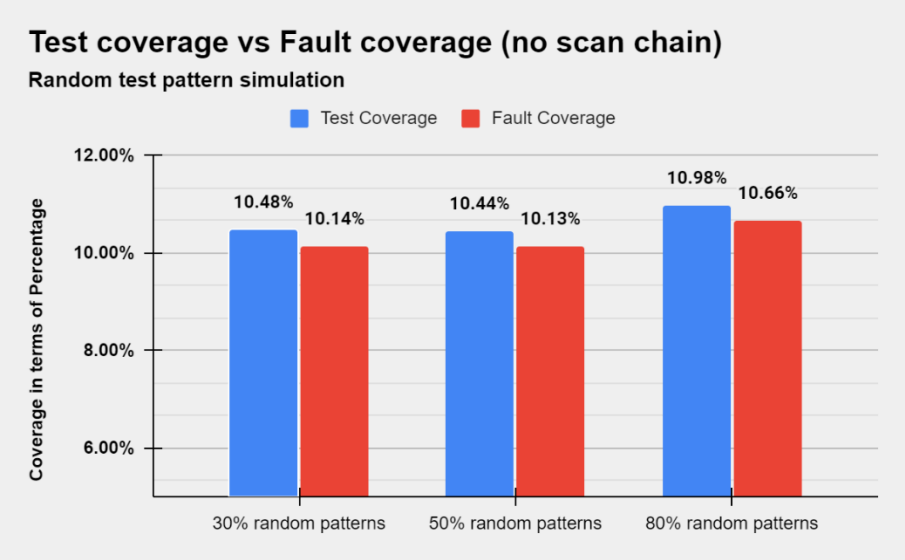

Fig 2.1.3: Graphical representation of Test coverage vs Fault coverage for no scan chain (30%,50%,80% patterns)

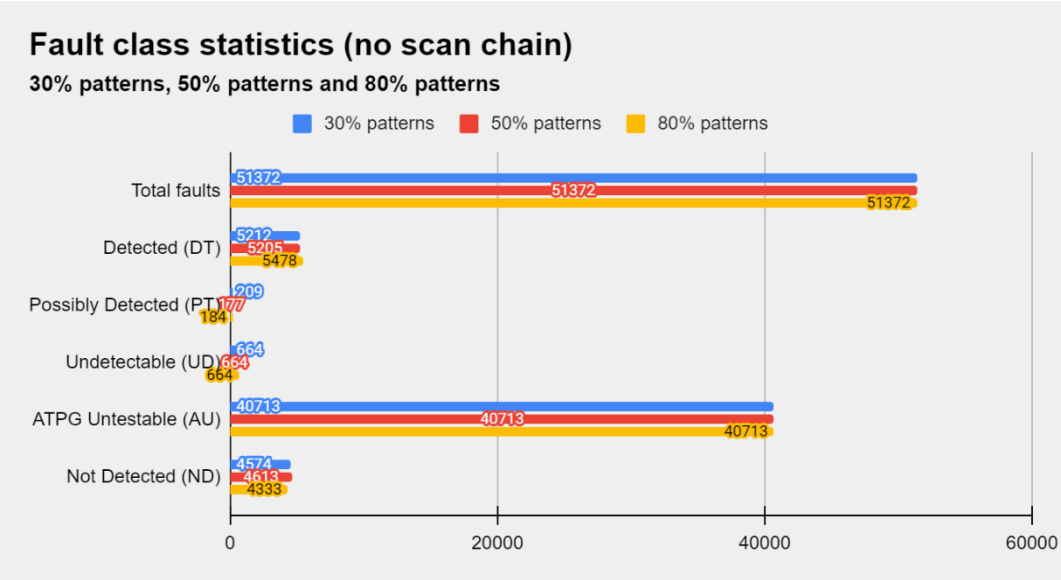

Fig 2.1.4: Graphical representation of Fault classes statistics for no scan chain (30%,50%,80% patterns)

SEQUENTIAL CIRCUIT (ONE SCAN CHAIN)

2.2.a) Deterministic Test Pattern Generation

Here using the Tetramax tool, we performed sequential test generation (one scan chain is involved) which is the full scan. The Verilog netlist file named “circuit.v” contains the D flipflops, in order to keep one scan chain we need to replace all d flip flops as scan flipflops. The new Verilog netlist named “circuit_sff.v” (under part2- 2a) in the attached zip file, consists additional inputs of scan input (SI) feeds the first flipflop, scan enable (SE) and additional output is scan out (SO) from last scan flip flop. To convert the “circuit.v” into the “circuit_sff” , we written a simple python code file named “……py” and made the necessary changes in the file.

Using the Tetramax, we generated the set of test patterns which detects all detectable faults in a given circuit. Here, to perform these operations we used the tetra max tool in shell by using the command “tmax -shell”. By launching the tetra max in shell mode, we run the “Test_Generation2a.tcl” which contains all the commands that requires to perform the required operations like creating a log file, reading netlists, adding clock, adding scan chains, running DRC, adding faults, and to generate patterns to a file.

After the ATPG is performed, there created the files called “test_gen2a.log”, “ATPG_pattern2a.pattern” files in the attached zip file (under part2-2a) which gives us the details of the faults and the patterns generated. From the output files, we observed the total faults, faults statistics, and number of patterns, Test coverage, and time taken by the tool as shown in below Table 2.2.1 and Table 2.2.2.

Table 2.2.1: Faults class statistics of Deterministic test pattern generation for full scan

Parameters		**Value**
Total Faults		54344
Fault classes	Detected (DT)	53150
	Possibly Detected (PT)	0
	Undetected (UD)	1192
	ATPG Untestable (AU)	2
	Not Detected (ND)	0

Table 2.2.2: Parameters of Deterministic test pattern generation for full scan

Parameters	**Value**
Total Faults	54344
Number of patterns generated	570
Test Coverage	100%
Fault Coverage	97.8%
Time taken by CPU	0.05s

From the above tables, we represented the observed data in the form of graphs as shown below. Fig 2.2.1 represents the comparison between the test coverage and fault coverage for full scan sequential circuit. Fig 2.2.2 represents the fault class statistics for full scan sequential circuit.

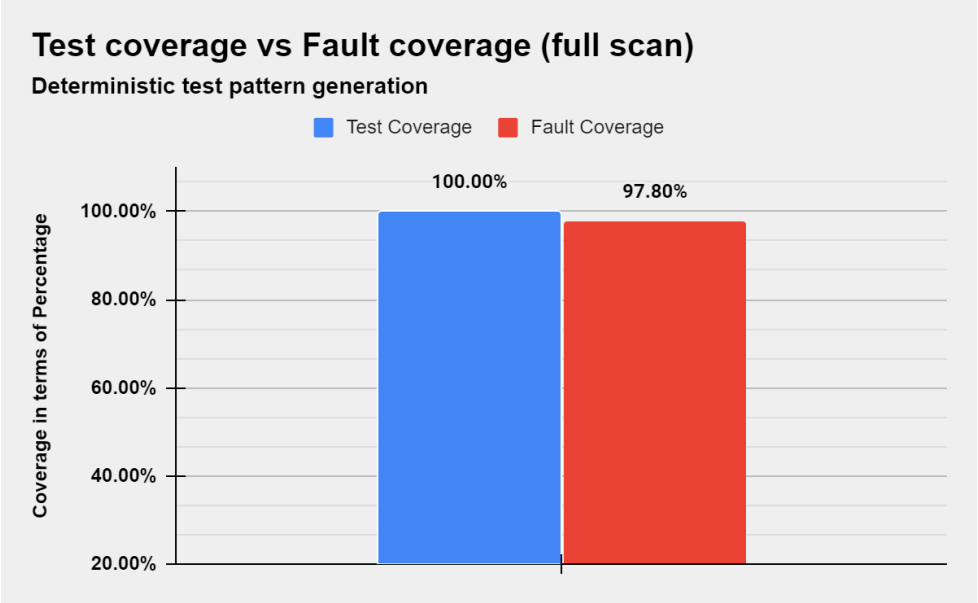

Fig 2.2.1: Graphical representation of Test coverage vs Fault coverage for Full scan

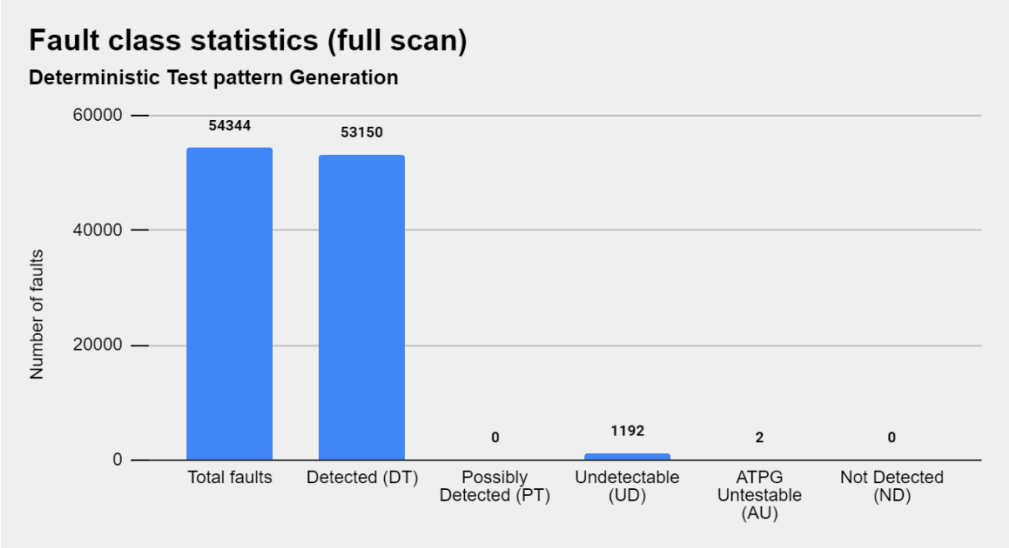

Fig 2.2.2: Graphical representation of Fault classes statistics for Full scan

2.2.b) Random Test Pattern Simulation

Here, we generate random patterns for both primary inputs and scan inputs i.e., 30% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 570 test patterns. So, 30% of those patterns is 172. Hence, we created the 172 random patterns by writing a python code. These generated random patterns are now used to generated '.pattern' file named "random_pattern_30". The '.tcl' file named "Test_Generation_fs_30" contains the

required commands to run the input pattern file and generate the "ATPG_pattern_fs_30". Now again we written another '.tcl' file named "Test_Generation_fl_30" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_30" and the fault left file is generated with the name of "faults_left_fl_30".

Table 2.2.3: Parameters of 30% Random pattern simulation for full scan

Parameters		**30% patterns**	**50% patterns**	**80% patterns**
		Value	**Value**	**Value**
Total Faults		54344	54344	54344
Fault classes	Detected (DT)	44651	46698	46716
	Possibly Detected (PT)	0	0	0
	Undetectable (UD)	133	133	133
	ATPG Untestable (AU)	0	0	0
	Not Detected (ND)	9560	7513	7495
Number of patterns generated		172	286	457
Test Coverage		82.37%	86.14%	86.17%
Fault Coverage		82.16%	85.93%	85.96%
Time taken by CPU		0.01s	0.01s	0.02s

2.2.c) Random Test Pattern Simulation

- Here, we generate random patterns for both primary inputs and scan inputs i.e., 50% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 570 test patterns. So, 50% of those patterns is 286. Hence, we created the 286 random patterns by writing a python code. These generated random patterns are now used to generated '.pattern' file named "random_pattern_50". The '.tcl' file named "Test_Generation_fs_50" contains the required commands to run the input pattern file and generate the "ATPG_pattern_fs_50". Now again we written another '.tcl' file named "Test_Generation_fl_50" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_50" and the fault left file is generated with the name of "faults_left_fl_50".

Table 2.2.4: Parameters of 50% Random pattern simulation for full scan

Parameters		**50% patterns**
		Value
Total Faults		54344
Fault classes	Detected (DT)	46698
	Possibly Detected (PT)	0
	Undetectable (UD)	133
	ATPG Untestable (AU)	0
	Not Detected (ND)	7513
Number of patterns generated		286
Test Coverage		86.14%
Fault Coverage		85.93%
Time taken by CPU		0.01s

- Again, we generated random patterns for both primary inputs and scan inputs i.e., 80% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 570 test patterns. So, 80% of those patterns is 457. Hence, we created the 457 random patterns by writing a python code. These generated random patterns are now used to generated '.pattern' file named "random_pattern_80". The '.tcl' file named "Test_Generation_fl_80" contains the required commands to run the input pattern file and generate the "ATPG_pattern_fs_80". Now again we written another '.tcl' file named "Test_Generation_fs_80" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_80" and the fault left file is generated with the name of "faults_left_fl_80".

Table 2.2.5 Parameters of 80% Random pattern simulation for full scan

Parameters		80% patterns Value
Total Faults		54344
Fault classes	Detected (DT)	46716
	Possibly Detected (PT)	0
	Undetectable (UD)	133
	ATPG Untestable (AU)	0
	Not Detected (ND)	7495
Number of patterns generated		457
Test Coverage		86.17%
Fault Coverage		85.96%
Time taken by CPU		0.02s

From the above tables, we represented the observed data in the form of graphs as shown below. Fig 2.2.3 represents the comparison between the test coverage and fault coverage for 30%, 50%,80% test patterns in full scan chain sequential circuit. Fig 2.2.4 represents the fault class statistics for 30%, 50%,80% test patterns in full scan chain sequential circuit.

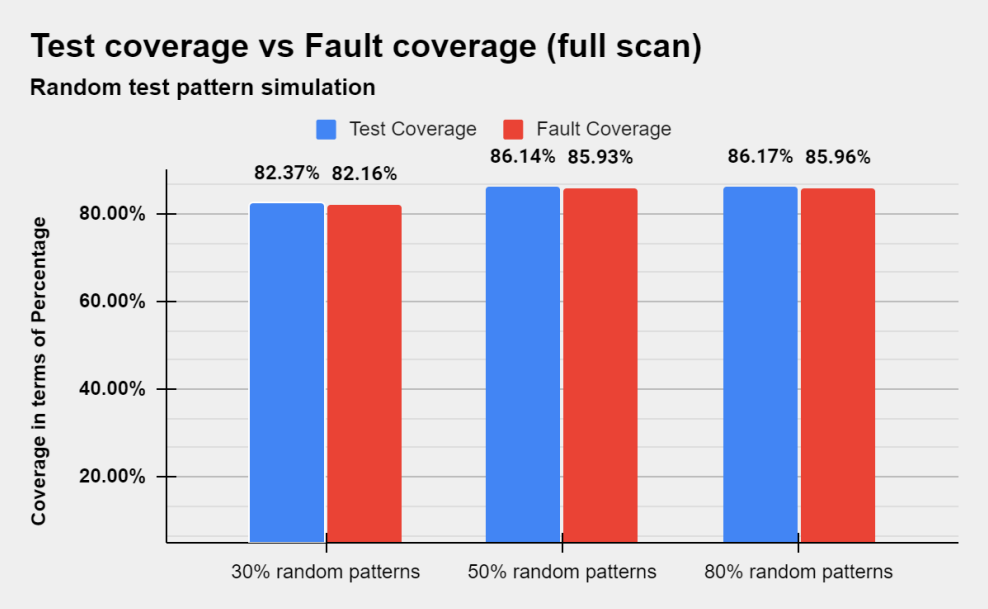

Fig 2.2.3: Graphical representation of Test coverage vs Fault coverage for full scan (30%, 50%, 80% patterns)

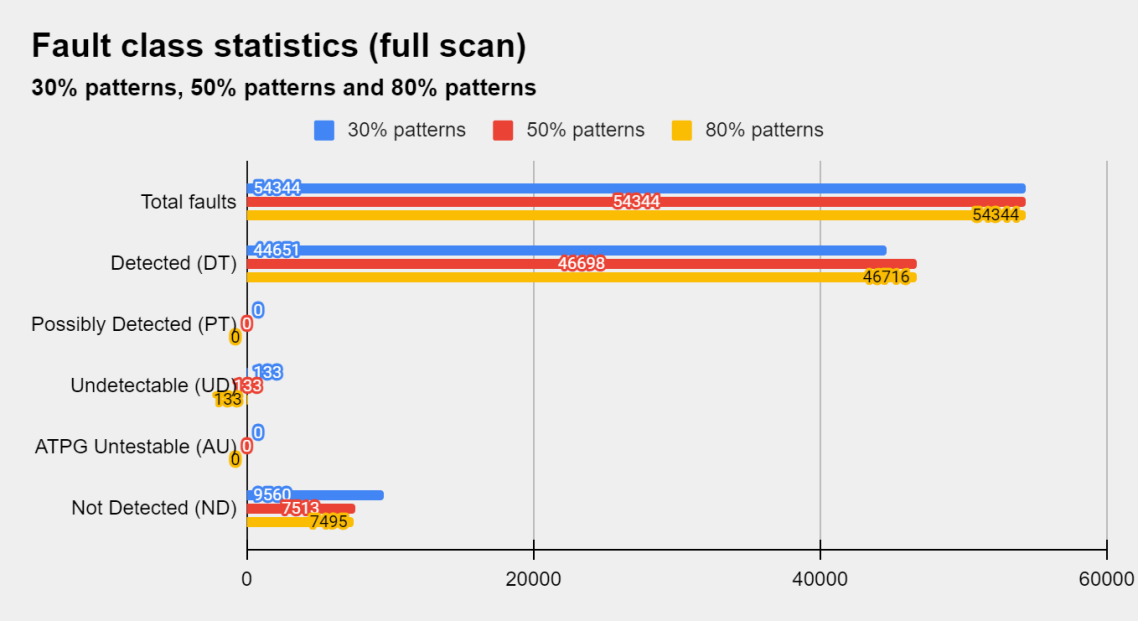

Fig 2.2.4: Graphical representation of Fault classes statistics for full scan (30%, 50%, 80% patterns)

SEQUENTIAL CIRCUIT (MULTIPLE SCAN CHAIN)

2.3.a) Deterministic Test Pattern Generation

Here using the Tetramax tool, we performed sequential test generation (multiple scan chain is involved). The Verilog netlist file named “circuit_sff.v” contains one scan chain, in order to keep multiple scan chain we need to distribute the scan flipflops into 10 scan chains. The new Verilog netlist named “circuit_sff.v” (under part3- 3a) in the attached zip file. This multiple scan chain netlist consists, 10 scan inputs and 10 scan outputs. The flipflops are distributed as four scan chains contains 54 flipflops each, and 6 scan chains contains 53 flipflops each.

Using the Tetramax, we generated the set of test patterns which detects all detectable faults in a given circuit. Here, to perform these operations we used the tetra max tool in shell by using the command “tmax -shell”. By launching the tetra max in shell mode, we run the “Test_Generation3a.tcl” which contains all the commands that requires to perform the required operations like creating a log file, reading netlists, adding clock, adding 10 scan chains, running DRC, adding faults, and to generate patterns to a file.

After the ATPG is performed, there created the files called “test_gen3a.log”, “ATPG_pattern3a.pattern” files in the attached zip file (under part3-3a) which gives us the details of the faults and the patterns generated. From the output files, we observed the total faults, faults statistics, and number of patterns, Test coverage, and time taken by the tool as shown in below Table 2.3.1 and Table 2.3.2.

Table 2.3.1: Faults class statistics of Deterministic test pattern generation for multiple scan

Parameters		**Value**
Total Faults		54416
Fault classes	Detected (DT)	53222
	Possibly Detected (PT)	0
	Undetected (UD)	1192
	ATPG Untestable (AU)	2
	Not Detected (ND)	0

Table 2.3.2: Parameters of Deterministic test pattern generation for multiple scan

Parameters	**Value**
Total Faults	54416
Number of patterns generated	576
Test Coverage	100%
Fault Coverage	97.8%
Time taken by CPU	0.06s

From the above tables, we represented the observed data in the form of graphs as shown below. Fig 2.3.1 represents the comparison between the test coverage and fault coverage for multiple scan chain sequential circuit. Fig 2.3.2 represents the fault class statistics for multiple scan chain sequential circuit.

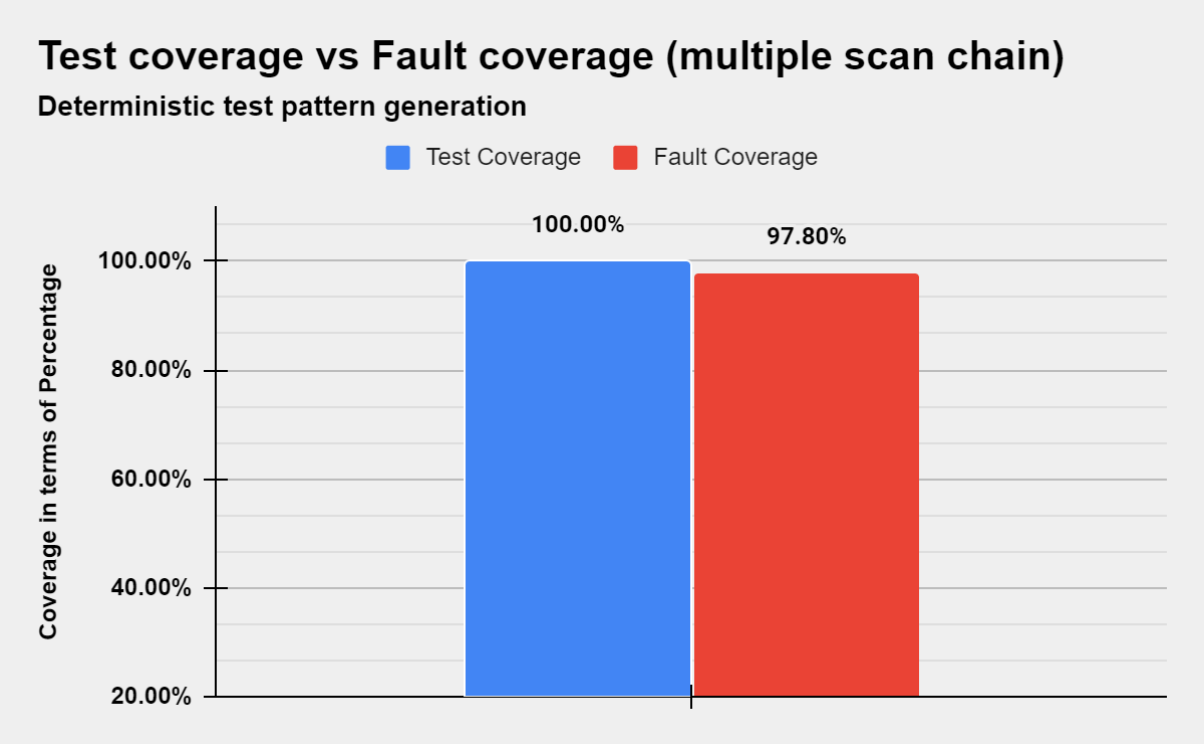

Fig 2.3.1: Graphical representation of Test coverage vs Fault coverage for Multiple scan chain

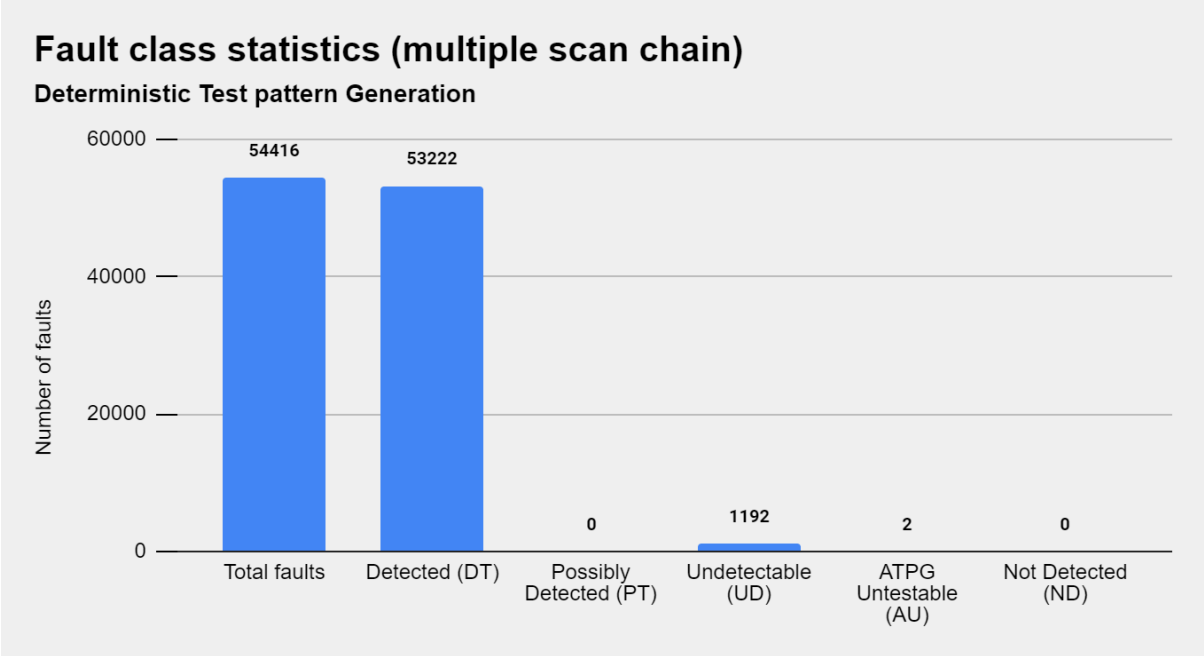

Fig 2.3.2: Graphical representation of Fault classes statistics for Multiple scan chain

2.3.b) Random Test Pattern Simulation

Here, we generate random patterns for both primary inputs and scan chain inputs i.e., 30% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 576 test patterns. So, 30% of those patterns is 174. Hence, we created the 174 random patterns by writing a python code. These generated random patterns are now used to generated '.pattern' file named "random_pattern_30". The '.tcl' file named "Test_Generation_fs_30" contains the required commands to run the input pattern file and generate the "ATPG_pattern_fs_30". Now again we written another '.tcl' file named "Test_Generation_fl_30" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_30" and the fault left file is generated with the name of "faults_left_fl_30".

Table 2.3.3: Parameters of 30% Random test pattern simulation for multiple scan chains

Parameters		**30% patterns**	**50% patterns**	**80% patterns**
		Value	**Value**	**Value**
Total Faults		54416	54416	54416
Fault classes	Detected (DT)	45549	45796	47392
	Possibly Detected (PT)	0	0	0
	Undetectable (UD)	133	133	133
	ATPG Untestable (AU)	0	0	0
	Not Detected (ND)	8734	8487	6891
Number of patterns generated		174	289	462
Test Coverage		83.91%	84.37%	87.31%
Fault Coverage		83.7%	84.15%	87.09%
Time taken by CPU		0.02s	0.01s	0.02s

2.3.c) Random Test Pattern Simulation

- Here, we generate random patterns for both primary inputs and scan chain inputs i.e., 50% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 576 test patterns. So, 50% of those patterns is 289. Hence, we created the 289 random patterns by writing a python code. These generated random patterns are now used

to generated '.pattern' file named "random_pattern_50". The '.tcl' file named "Test_Generation_fs_50" contains the required commands to run the input pattern file and generate the "ATPG_pattern_fs_50". Now again we written another '.tcl' file named "Test_Generation_fl_50" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_50" and the fault left file is generated with the name of "faults_left_fl_50".

Table 2.3.4: Parameters of 50% Random test pattern simulation for multiple scan chains

Parameters		**50% patterns**
		Value
Total Faults		54416
Fault classes	Detected (DT)	45796
	Possibly Detected (PT)	0
	Undetectable (UD)	133
	ATPG Untestable (AU)	0
	Not Detected (ND)	8487
Number of patterns generated		289
Test Coverage		84.37%
Fault Coverage		84.15%
Time taken by CPU		0.01s

- Here, we generate random patterns for both primary inputs and scan chain inputs i.e., 80% patterns of the number of patterns generated in the deterministic test pattern generation. There we got 576 test patterns. So, 80% of those patterns is 462. Hence, we created the 462 random patterns by writing a python code. These generated random patterns are now used to generated '.pattern' file named "random_pattern_80". The '.tcl' file named "Test_Generation_fs_80" contains the required commands to run the input pattern file and generate the "ATPG_pattern_fs_80". Now again we written another '.tcl' file named "Test_Generation_fl_80" to perform the fault simulation. After the fault simulation we observed the fault class statistics in the generated log file named "test_gen_fl_80" and the fault left file is generated with the name of "faults_left_fl_80".

Table 2.3.5: Parameters of 80% Random test pattern simulation for multiple scan chains

Parameters		80% patterns Value
Total Faults		54416
Fault classes	Detected (DT)	47392
	Possibly Detected (PT)	0
	Undetectable (UD)	133
	ATPG Untestable (AU)	0
	Not Detected (ND)	6891
Number of patterns generated		462
Test Coverage		87.31%
Fault Coverage		87.09%
Time taken by CPU		0.02s

From the above tables, we represented the observed data in the form of graphs as shown below. Fig 2.3.3 represents the comparison between the test coverage and fault coverage for 30%, 50%,80% test patterns in multiple scan chain sequential circuit. Fig 2.3.4 represents the fault class statistics for 30%, 50%,80% test patterns in multiple scan chain sequential circuit.

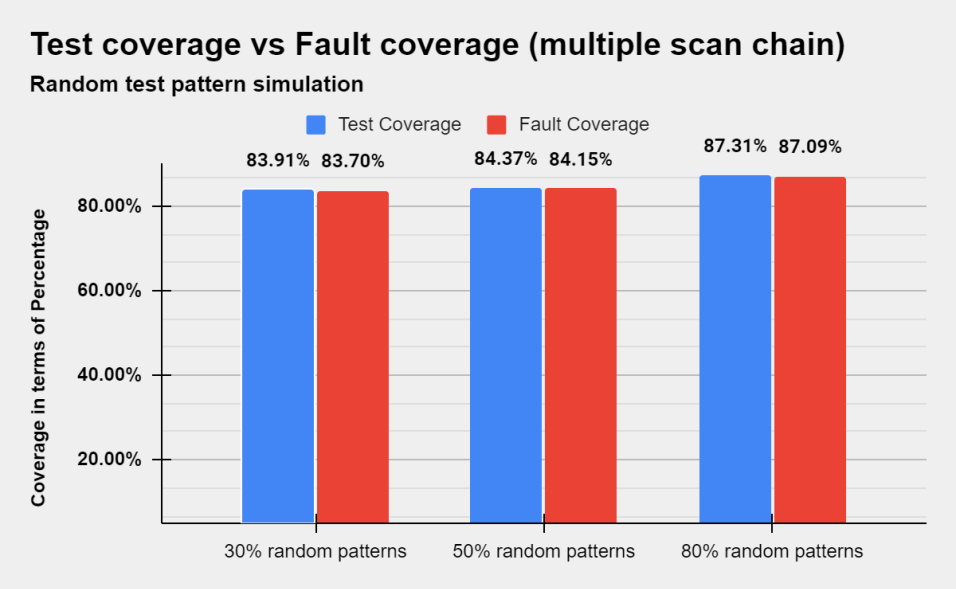

Fig 2.3.3: Graphical representation of Test coverage vs Fault coverage for multiple scan chain (30%, 50%, 80% patterns)

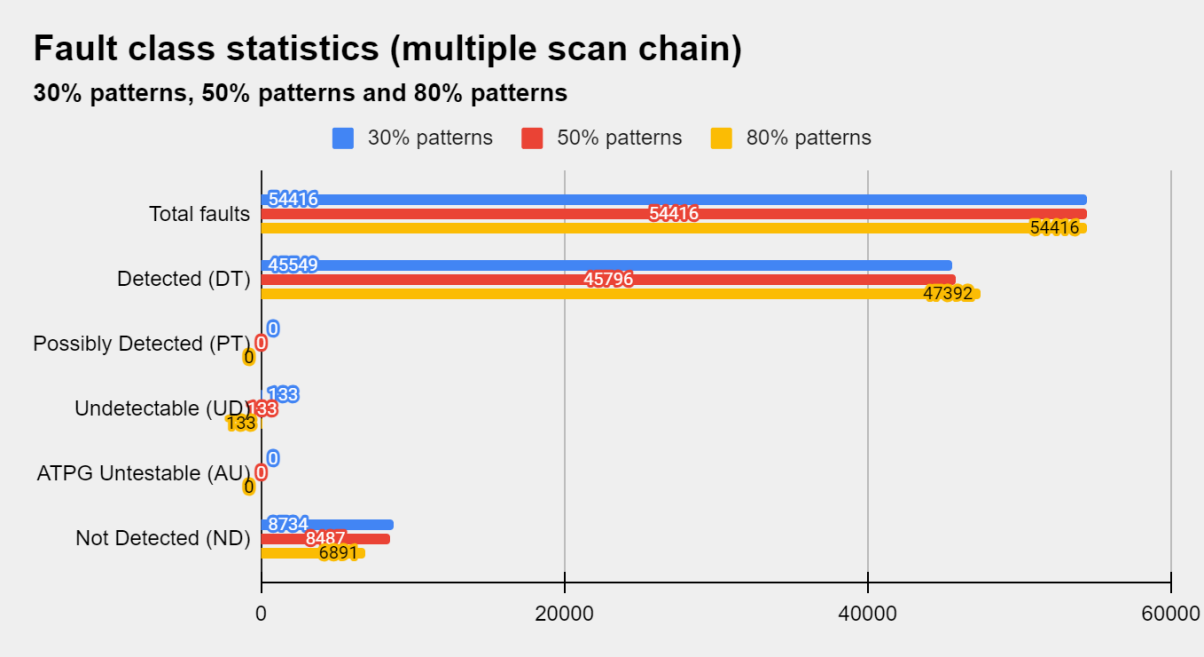

Fig 2.3.4: Graphical representation of Fault classes statistics for multiple scan chain (30%, 50%, 80% patterns)

SEQUENTIAL CIRCUIT (PARTIAL SCAN)

2.4.a) Deterministic Test Pattern Generation (30% flip flops in scan)

Here using the Tetramax tool, we performed sequential test generation (partial scan is involved). The Verilog netlist file named “circuit.v” contains all D flip flops in order make it as partial scan, here we converted 30% of flip flops randomly into the scanned flip flops. The new Verilog netlist named “circuit_sff.v” (under part4- 4a) in the attached zip file. The partial scan consists of both D flip flops and scan flip flops hence we included two different clocks for the both d and scan flip flops. The partial scan operates in two modes i.e., in normal mode (both clocks are active) and in scan mode (only scan flip flops clock is active). To make these necessary changes we written a python code shown in file named “……py” and generated the partial scan Verilog netlist.

Using the Tetramax, we generated the set of test patterns which detects all detectable faults in a given circuit. Here, to perform these operations we used the tetra max tool in shell by using the command “tmax -shell”. By launching the tetra max in shell mode, we run the “Test_Generation4a.tcl” which contains all the commands that requires to perform the required operations like creating a log file, reading netlists, adding clock, adding scan chains, running DRC, adding faults, and to generate patterns to a file.

After the ATPG is performed, there created the files called “test_gen4a.log”, “ATPG_pattern4a.pattern” files in the attached zip file (under part4-4a) which gives us the details of the faults and the patterns generated. From the output files, we observed the total faults, faults statistics, and number of patterns, Test coverage, and time taken by the tool as shown in below Table 2.4.1 and Table 2.4.2.

Table 2.4.1: Faults class statistics of Deterministic test pattern generation for partial scan1

Parameters		**Value**
Total Faults		52132
Fault classes	Detected (DT)	40953
	Possibly Detected (PT)	398
	Undetected (UD)	1199
	ATPG Untestable (AU)	4738
	Not Detected (ND)	4844

Table 2.4.2: Parameters of Deterministic test pattern generation for partial scan1

Parameters	**Value**
Total Faults	52132
Number of patterns generated	458
Test Coverage	80.80%
Fault Coverage	78.55%
Time taken by CPU	32064s (8:54hrs)

2.4.b) Deterministic Test Pattern Generation (another 30% flip flops in scan)

Here using the Tetramax tool, we performed sequential test generation (partial scan is involved). The Verilog netlist file named "circuit.v" contains all D flip flops in order make it as partial scan, here we converted another 30% of flip flops (different from 4.a) randomly into the scanned flip flops. The new Verilog netlist named "circuit_sff.v" (under part4- 4b) in the attached zip file. This netlist also generated with all changes which mentioned in part 4.a. To make these necessary changes we written a python code shown in file named "……py" and generated the partial scan Verilog netlist.

Using the Tetramax, we generated the set of test patterns which detects all detectable faults in a given circuit. Here, to perform these operations we used the tetra max tool in shell by using the command "tmax -shell". By launching the tetra max in shell mode, we run the "Test_Generation4b.tcl" which contains all the commands that requires to perform the required operations like creating a log file, reading netlists, adding clock, adding scan chains, running DRC, adding faults, and to generate patterns to a file.

After the ATPG is performed, there created the files called "test_gen4b.log", "ATPG_pattern4b.pattern" files in the attached zip file (under part4-4b) which gives us the details of the faults and the patterns generated. From the output files, we observed the total faults, faults statistics, and number of patterns, Test coverage, and time taken by the tool as shown in below Table 2.4.3 and Table 2.4.4.

Table 2.4.3: Faults class statistics of Deterministic test pattern generation for partial scan2

Parameters		**Value**
Total Faults		52680
Fault classes	Detected (DT)	45660
	Possibly Detected (PT)	440
	Undetected (UD)	1196
	ATPG Untestable (AU)	3056
	Not Detected (ND)	2328

Table 2.4.4: Parameters of Deterministic test pattern generation for partial scan 2

Parameters	**Value**
Total Faults	52680
Number of patterns generated	386
Test Coverage	89.12%
Fault Coverage	86.67%
Time taken by CPU	19801s (5:30hrs)

From the above tables, we represented the observed data in the form of graphs as shown below. Fig 2.4.1 represents the comparison between the test coverage and fault coverage for partial scan sequential circuit. Fig 2.4.2 represents the fault class statistics for partial scan sequential circuit.

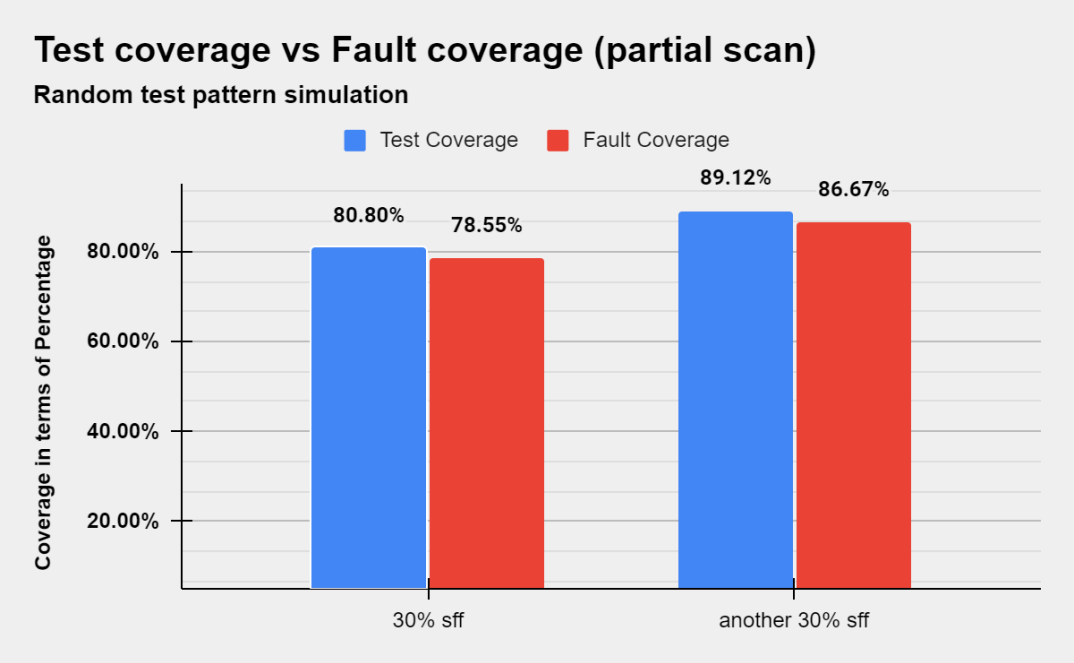

Fig 2.4.1: Graphical representation of Test coverage vs Fault coverage for partial scan

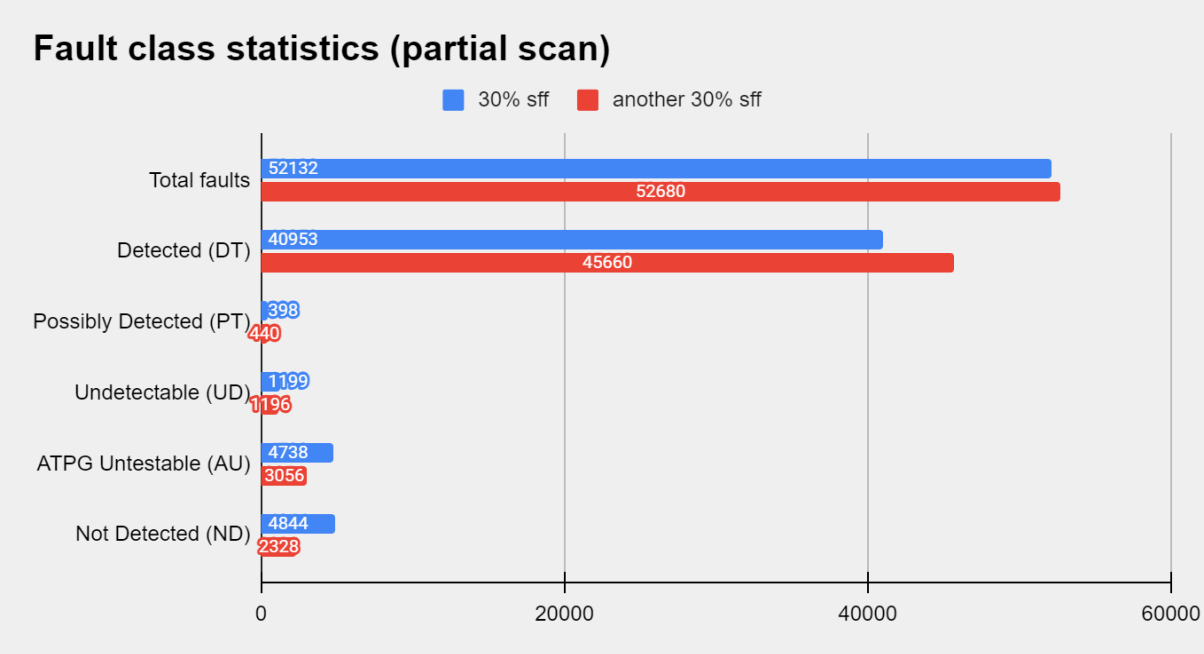

Fig 2.4.2: Graphical representation of Fault classes statistics for partial scan

COMPARISON BETWEEN DIFFERENT SCAN METHODS

In this section, we compared every technique used in this project in testing and simulating the sequential circuit. The methods used are without scan chain, full scan, multiple scan chain, and partial scan. Now we compared the test coverage of each technique and represented them in a graphical format as shown in Fig 3.1. We observed that the test coverage is 100% for both full scan and multiple scan techniques. The lowest test coverage is seen with no scan chain technique.

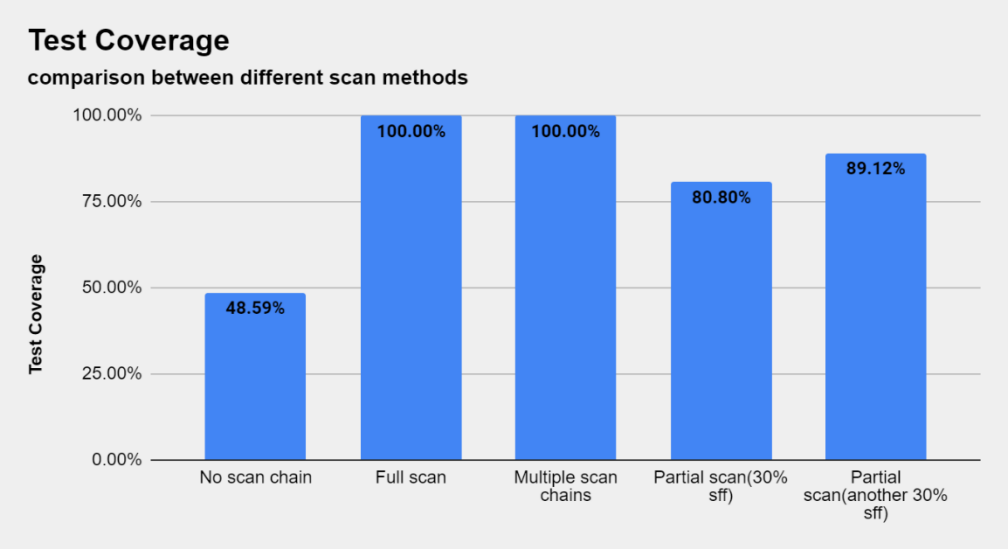

Fig 3.1: Comparison of test coverage between different scan techniques

Now, we compared the fault coverage of each technique and represented them in a graphical format as shown in Fig 3.2. We observed that the fault coverage is high and same i.e., 97.8% for both full scan and multiple scan techniques. The lowest fault coverage is seen with no scan chain technique.

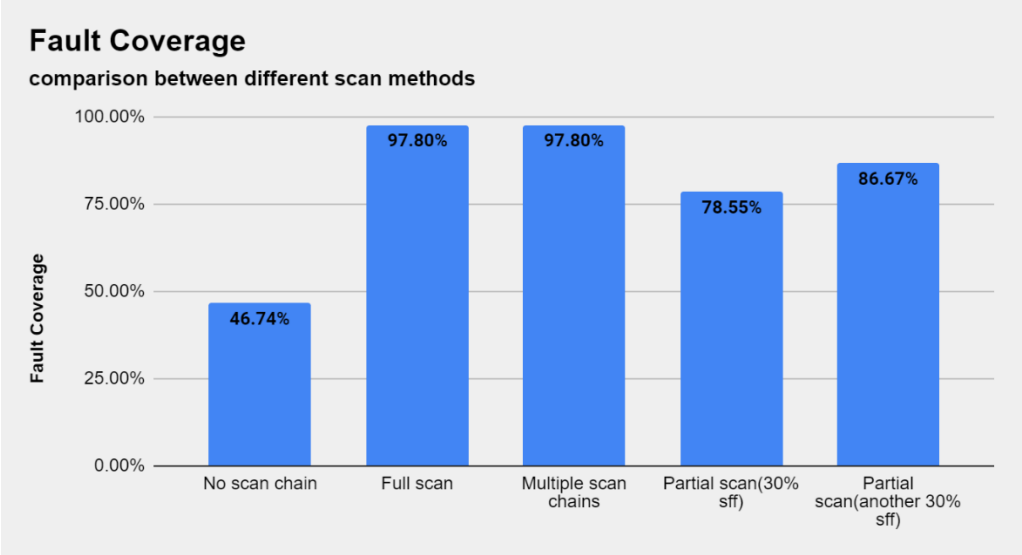

Fig 3.2: Comparison of fault coverage between different scan techniques

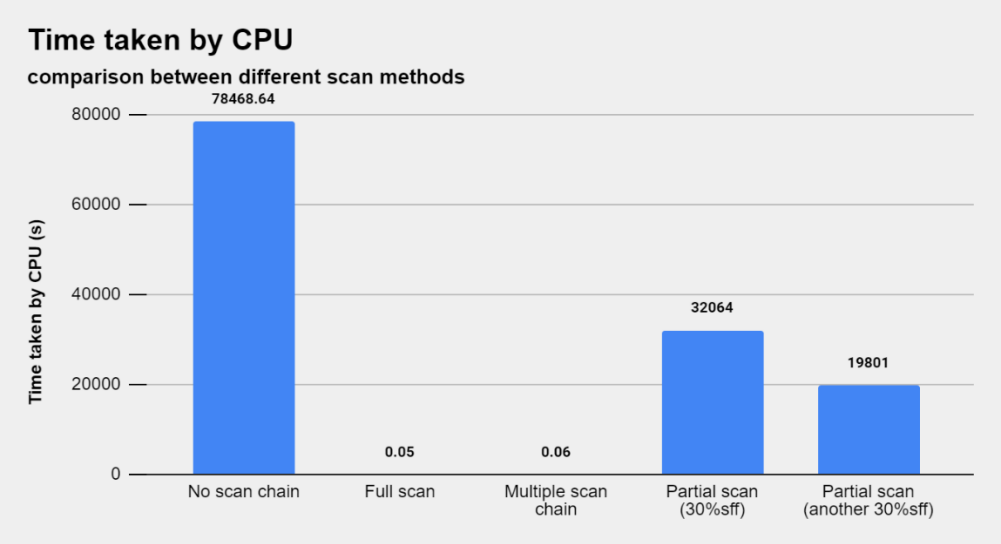

Fig 3.3: Time taken by CPU in different scan techniques

CONCLUSION

The main aim of the project is to get familiar with all scan techniques used to test and simulate the sequential circuits using ATPG tool i.e., tetramax. Here starting with no scan chain in the sequential circuit, to full scan, to multiple scan chains, and to partial scan technique we performed deterministic test generation and also performed the random pattern simulation. We written .tcl files with the required commands to perform the operations and generated the test patterns and also fault simulated with random patterns generated. Finally, we compared each technique test coverage, fault coverage and time taken by CPU. Hence, we concluded that both Full scan technique and Multiple scan chain technique gives us the best results by considering the factors of Test coverage, Fault Coverage and Time taken by CPU. At the end of the project, we are able to understand the usage of ATPG tool and performed the project successfully.