VHDL SWITCH-LEVEL FAULT SIMULATION

Christopher A. Ryan
Department of Electrical Engineering
University of Kentucky
Lexington, KY 40506-0046
email: cryan@engr.uky.edu

Joseph G. Tront
Bradley Department of Electrical Engineering
Virginia Polytechnic Institute&State University
Blacksburg, VA 24061-0111
email: jgtront@vtvm1.cc.vt.edu

Abstract

Switch level faults, as opposed to traditional gate level faults, can more accurately model physical faults found in an integrated circuit. This paper presents a novel VHDL switch level fault simulator. The simulator implementation uses 9-valued logic, N and P-type switch state tables, a minimum operation, and a switch level extension to parallel fault single pattern fault simulation. The advantages of this technique and hardware description language implementation are fault modeling accuracy and a straight forward implementation that compiles switches and faults to VHDL code.

1. Introduction

One problem with performing fault simulation at the gate level is that it has been shown that physical faults within a metal oxide semiconductor (MOS) IC cannot be modeled by only gate level stuck-at faults [Shen85]. Switch level fault simulation has greater accuracy and can model physical faults better.

The simulators reported on in [Bose82], [Kawai84], [Bryant85], [Barzi86], [Shih86], [Bryant87], [Lee91] and [Meyer93], are all switch-level fault simulators. These simulators uses a variety of modeling and simulation techniques. The modeling ranges from representing the transistor as a voltage controlled current source to using a strength system for transistor and nodes to converting the circuit to a set of Boolean state matrix equations.

This paper presents a novel compiled code VHDL switch level fault simulator. Also presented here is the switch level extension to parallel fault single pattern (PFSP) fault simulation. The advantages of this technique and implementation are that it has the fault modeling accuracy of switch level simulators such as CSASIM [Kawai84] and FMOSSIM [Bryant85], and that it has a straight forward compiled type simulator implementation such as COSMOS [Bryant87]. To preserved fault modeling accuracy, the PFSP switch level fault simulation *mask* operation is used to inject faults. While faults are injected at the individual switch, the *mask* operation is internal to the switch. The compiled type simulator is maintained because all switches (and injected faults) are compiled to VHDL code.

The remainder of this paper is outlined as follows. Section 2 gives an overview of the switch level models, the mathematics, and fault models used for fault simulation. The switch level extension to PFSP fault simulation is given in Section 3. Section 4 describes the novel VHDL fault simulator. Fault simulation results for benchmark circuits are also given in Section 4. Section 5 gives the summary.

2. Switch Level Models and Algebra

This research uses the IEEE standard 1164 nine-valued logic [Bill91] for switch level fault simulation, similar to that used by [Hayes82], and [Hayes87]. The values, listed here ranging from the weakest to the strongest, are {Z, -, L, H, W, 0, 1, X, U}. The values of 1 and 0 represent logic levels forcing high and low, respectively. The values of H and L represent logic levels weak high and low respectively. The values of X and W represent forcing and weak unknown respectively. The values of U, -, and Z represent uninitialized, don't care, and high impedance, respectively.

Using nine-valued logic, the N - type MOS switch is modeled by a state table as shown in Table 2.1 where the input is applied to the drain and the output is taken at the source. Shown in Figure 2.1, the transistor switch model has basically three modes; off, on, and unknown.

Table 2.1 N-type switch state table.

Present State	Input	Gate	Next State
-	0	1,H	0
-	1	1,H	H
-	H	1,H	W
-	L	1,H	L
-	U,X,W,-	1,H	U,X,W,-
-	U,X,0	U,X,Z,W,-	X
-	1,W,H,L,-,Z	U,X,W,-,Z	W
-	Z	1,H	W
U	-	0,L	U
X	-	0,L	W
1	-	0,L	H
0	-	0,L	L
W,H,L,-,Z	-	0,L	Z

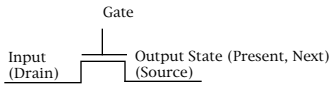

Figure 2.1 Switch model (N-type).

In the off mode, the N - type switch isolates the output from the input. The next output value is then based

on the previous output value and the capacitance at the output node. Since that output node is isolated from both power and ground, it is assumed that while the transistor is off a previous output value of a forcing 1 degrades to a weak H, a forcing 0 to a weak L, a forcing X (1 or 0) to a weak W (H or L), and all other weak W, H, L, -, Z to high impedance Z.

In the on mode, the N - type switch connects input to output and effectively passes the input value to the output. However, assuming the worst-case, the N - type switch degrades the 1 and H values to H and W, respectively.

In the unknown mode, the N - type switch can be either on or off. Thus in the worst-case, the output becomes forcing unknown X with strong inputs U, X, or 0 present, while the output becomes weak unknown W with weak inputs 1, W, H, L, -, or Z present.

Not shown here, the P - type MOS switch is modeled similarly to the N - type MOS switch except its on and off modes are complimentary to that of the N - type switch. Also in worst - case situations, the P - type switch degrades the 0 and L values to L and W values, respectively.

The nine-valued logic uses operators AND, OR, NOT, and RESOLVE as described in [Bill91]. Repeated here in Table 2.2, the RESOLVE operator will be used to implement the nine-valued CONNECTOR (*) operator. The CONNECTOR is used to resolve fan - in signals at the nodes. In addition, the MINIMUM (MIN) operator as shown in Table 2.3 will be used. Both the CONNECTOR and the MINIMUM operators are used for the PFSP switch level fault simulation as described in Section 3.

Table 2.2 Nine-valued connector operation.

*	U	X	0	1	Z	W	L	H	-
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	X	0	0	0	0	0
1	U	X	X	1	1	1	1	1	1
Z	U	X	0	1	Z	W	L	H	-
W	U	X	0	1	W	W	W	W	W
L	U	X	0	1	L	W	L	W	L
H	U	X	0	1	H	W	W	H	H
-	U	X	0	1	-	W	L	H	-

Table 2.3 Minimum operation.

Min	U	X	0	1	Z	W	L	H	_
U	U	X	0	1	Z	W	L	H	-
X	X	X	0	1	Z	W	L	H	-
0	0	0	0	0	Z	W	L	H	-
1	1	1	0	1	Z	W	L	H	-
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
W	W	W	W	W	Z	W	L	H	-
L	L	L	L	L	Z	L	L	L	-
H	H	H	H	H	Z	H	L	H	-
-	-	-	-	-	Z	-	-	-	-

The fault set considered in this research is line stuck-at-1/0 on the switch. Transistor stuck on / off faults are modeled by the line stuck-at-1/0 on the gate of the switch. Only a single fault per circuit is assumed, i.e., the classic single fault model is used. While the fault simulator type is compiled, in this research fault injection is used for the line stuck-at faults. The line stuck-at faults are injected via a *mask* in parallel fault simulation. More detail on the injection of line stuck-at faults will be given in Section 3.

3. Parallel Fault Simulation

3.1 Introduction

Parallel fault single pattern fault simulation is well-established for the gate level [Fuji85]. This section discusses the parallel fault simulation technique as extended to the switch level. Included in this discussion is an example from the VHDL simulation results of a single N - type switch.

3.2 Switch Level Parallel Fault Simulation

As shown in Figure 3.1, parallel fault simulation is extended to the switch level by using the nine-valued logic and the N-type and P-type switch state tables described in Section 2. During parallel fault simulation, faults are injected on each signal line by the MASK blocks. Two constant vectors used in the MASK procedure are the *fvalue* vector (GF,DF,SF) and the *mask* vector (GM,DM,SM) as shown in Tables 3.1 and 3.2, respectively.

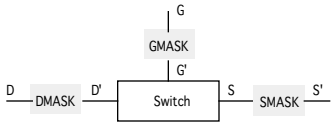

Figure 3.1 Switch Level Parallel Fault Simulation.

Table 3.1 Constant fvalue vectors.

	[FF	G1	G0	D1	D0	S1	S0]
GF	[Z	1	0	Z	Z	Z	Z]
DF	[Z	Z	Z	1	0	Z	Z]
SF	[Z	Z	Z	Z	Z	1	0]

Table 3.2 Constant mask vectors.

	[FF	G1	G0	D1	D0	S1	S0]
GM	[U	Z	Z	U	U	U	U]
DM	[U	U	U	Z	Z	U	U]
SM	[U	U	U	U	U	Z	Z]

The *mask* vector is used to inject the faults and has value of Z for faulty positions and U at all other positions. Since Z has the lowest strength and U has the highest strength of the values in the nine-value logic, the minimum operation using the *mask* vector and the good signal vectors as inputs will always yield a Z in the faulty positions.

The *fvalue* vector is used to select the fault value and contains a position for the fault free value and for each faulty value. Here, a line with fault stuck-at-1 has value 1, while a line with fault stuck-at-0 has value 0. All other values are Z. For example, the transistor gate *fvalue* vector GF has a value of 1 in the G1 position designating a stuck-at-1 type fault on the gate, while there is a 0 in the G0 position designating a stuck-at-0 type fault. Since all other values are Z, the minimum operation is now used in order to select the faults 1 and 0 in the G1 and G0 positions, while all other positions remain at their good value.

The Bold face constants in Tables 3.1, 3.2 and in Equations 3.1 to 3.7 are defined as follows: **G -** Gate input **D -** Drain input **S -** Source output, **S-1 -** Previous Source output, **GF -** Gate *fvalue* vector, **DF -** Drain *fvalue* vector, **SF -** Source *fvalue* vector, **GM -** Gate *mask* vector, **DM -** Drain *mask* vector, **SM -** Source *mask* vector, **FF -**

Fault Free Position, **G1 -** Gate Stuck-At-1 Position, **G0 -** Gate Stuck-At-0 Position, **D1 -** Drain Stuck-At-1 Position, **D0 -** Drain Stuck-At-0 Position, **S1 -** Source Stuck-At-1 Position, **S0 -** Source Stuck-At-0 Position,

In order to perform switch vector operations, the gate, the drain, and the previous source good values must be expanded to the word length as

$$G <= [GGGGGGG], \quad (3.1)$$

$$D <= [DDDDDDD], \quad (3.2)$$

and

$$S_{-1} <= [S_{-1}S_{-1}S_{-1}S_{-1}S_{-1}S_{-1}S_{-1}], \quad (3.3)$$

respectively, where G, D, and S_{-1} are the good values.

Next, in order to inject faults at the inputs G and D, the following switch level *mask* operations are performed as

$$G' = [MIN(G,GM)]*GF \quad (3.4)$$

and

$$D' = [MIN(D,DM)]*DF, \quad (3.5)$$

where * is the CONNECTOR function, and MIN() is the MINIMUM function. Remember that GF, DF, GM, and DM are all vectors and to calculate G' and D' vector operations must be performed. The output of the switch is determined by

$$S = SWITCH[G,D,S_{-1}], \quad (3.6)$$

where SWITCH is the switch function for the N-type or P-type transistor, and S_{-1} is the previous output value of the switch. In order to inject faults on the output, the *mask* operation is performed as

$$S' = [MIN(S,SM)]*SF, \quad (3.7)$$

where S is defined in Equation 3.6. The resulting output vector contains positions for the fault-free output as well as positions for all outputs in the presence of each specific fault. Each fault is considered detected if that fault position output is noticeably different from the fault-free position output. Noticeably different is defined as being a result of a 1 or an H instead of a 0 or an L. Output values of W, and X and not considered noticeably different.

An example of switch level parallel fault simulation is given below. The results for this example, shown as the SOURCE value after the output MASK operation, indicate that the three faults gate stuck-at-0, drain stuck-at-1, and source stuck-at-1 are detected for the test vector, while the three faults gate stuck-at-1, drain stuck-at-0, and source stuck-at-1 are not detected. The input test vector for this example is <D,G> = <0,1>, with the previous source value S_{-1} equal to 1.

First, use Equations 3.1 - 3.3 in order to expand gate, drain and previous source values as

INPUT
SOURCE_P (value = 1) = (1111111) (3.8)
DRAIN (value = 0) = (0000000) (3.9)
GATE (value = 1) = (1111111) (3.10)

Second, perform *mask* operation on the expanded drain and gate using Equations 3.4 and 3.5 as

MASK OPERATION
DRAIN'(value = "0001000") (3.11)
GATE' (value = "1101111") (3.12)

Next, perform the switch operation using Equations 3.6 as

SWITCH OPERATION
SOURCE' (value = "00HH000") (3.13)

Finally, perform the output *mask* operation using Equation 3.7 as

MASK OPERATION
SOURCE (value = "00HH010") (3.14)

Equation 3.14 gives the resulting vector that includes the fault free output and outputs in the presence of the stuck-at fault set. Since the fault free output is 0, and since there is an H in positions G0 and D1, the faults gate stuck-at-0 and drain stuck-at-1 are detected for the applied input vector . Similarly, since there is a 1 in positions S1, the fault source stuck-at-1 is detected. Faults not detected are the faults of the positions whose values match that of the fault free output and have value 0.

The PFSP fault simulation technique as described above was verified for the 9^3 combinations of inputs and previous states using a VHDL simulator [Synop90]. Using the PFSP fault simulation technique, the CONNECTOR operation and preprocessing, switch level circuits can be compiled to VHDL code and then fault simulated using a VHDL simulator.

4. VHDL Switch Level Fault Simulation

4.1 Introduction

A block diagram of the switch level fault simulator is shown in Figure 4.1. First, a switch level netlist is extracted from layout. Second, the switch netlist is preprocessed to produce a reverse level ordered netlist. Next, the reverse level ordered switch netlist is compiled to VHDL code as parallel switches (PSs) and nodes (NODESs). Each PS performs PFSP fault simulation for one switch. Each NODES performs the CONNECTOR (*) operation for each node. Finally, fault simulation is performed using the controller or VHDL Testbench and the user input test vectors.

4.2 Circuit Netlist Partitioning

The circuit netlist partitioning used here is reverse level ordering. The reverse level ordering of an example XNOR gate is shown in Figure 4.2. The ordering process starts at the primary outputs and stops at the primary inputs. When a fan - in node is transverse all transistors that fan - in to that node are placed in that same reverse level.

4.3 VHDL Fault Simulator

This section describes the components that make up the fault simulator. The components include the PS, the NODES, and the controller. The PSs and NODESs structure is compiled from the reverse level ordered netlist. One transistor in the reverse level ordered netlist is mapped to one PS in the VHDL netlist. The nodes in the reverse level ordered netlist are mapped to NODESs in the VHDL netlist.

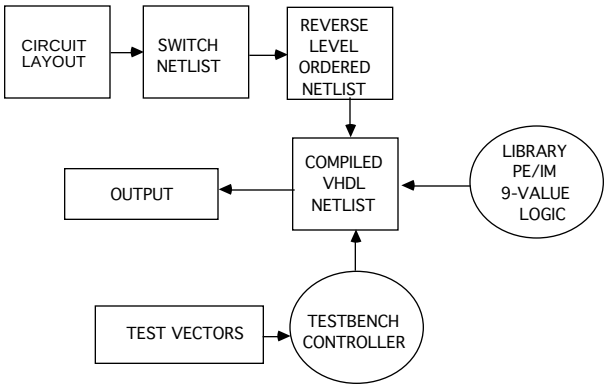

Figure 4.1 Fault simulation verification block diagram.

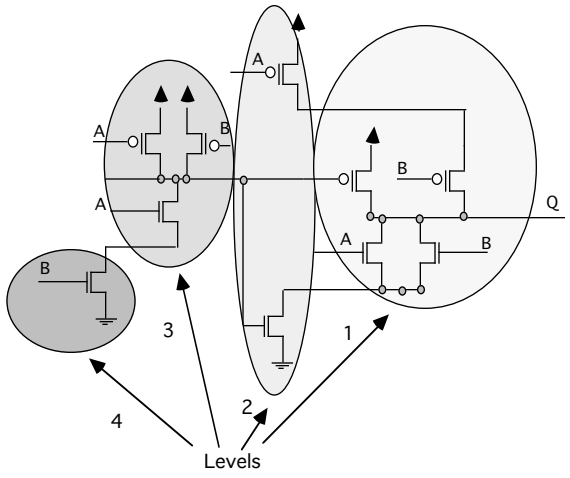

Figure 4.2 Reverse Level Ordering of CMOS XNOR gate.

Each PS is used to model one switch of the circuit and performs PFSP fault simulation using the technique described in section 3. The four modes of operation of the PS, as used in fault simulation, are shown in Figure 4.3. The good simulation and the inject simulation modes model one non-faulty N - type or P - type switch. The difference in these modes is that good simulation is used on the good test vector while inject simulation is used to propagate faulty values. Parallel fault simulation mode simulates all switch level faults for one N - type or P - type switch. Fault inject mode is used to inject faults into the simulated circuit. Finally, each PS keeps a table of its own faults as detected or undetected.

As shown symbolically in Figure 4.4, the NODE is used to resolve fan-in at a node in the circuit. Signal values input from lines C1, C2, ..., CN are resolved to output value C0 using the nine - value connector operation. The CONNECTOR operation is the nine-value resolve operator from the IEEE standard 1164 nine-value logic [Bill91]. Since VHDL is designed for hardware modeling the CONNECTOR is TYPE defined. TYPE defined means that a variable or signal which includes an operation can be defined. In the case of the CONNECTOR operation, the gate, the drain and the source signals are all defined with the CONNECTOR type.

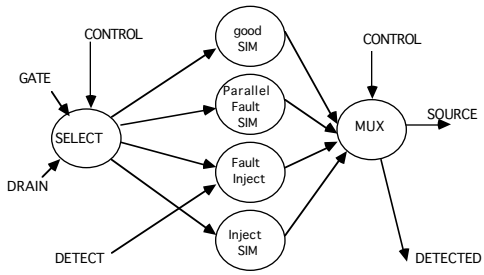

Figure 4.3 Parallel Switch.

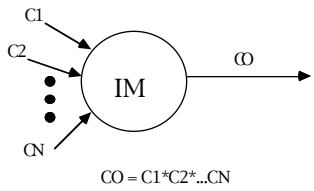

Figure 4.4 NODE.

The controller shown in Figure 4.5 is used to model the parallel fault simulation algorithm. The controller is the VHDL stimulus file or Testbench and provides the control signals for the simulated PSs by sequencing through the four modes described earlier. The Controller starts with good circuit simulation. This is followed by fault simulation for one reverse level of switches. The potentially detected faults are, next, injected one at a time. These potentially detected faults are then simulated to the output or dropped when undetected. The fault simulation includes fault dropping of detected faults.

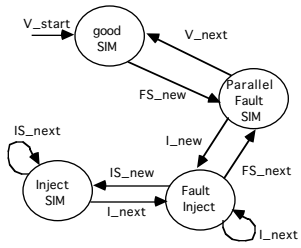

Figure 4.5 Testbench controller.

Using the switch level extension to parallel fault simulation, all faults for one switch are simulated in parallel with one PS. Faults detected in the level are propagated to the primary output using switch level

implication [Light82]. The fault simulation algorithm is shown in the flowchart in Figure 4.6. First, the reverse level of faults that is to be fault simulated, is initialized to reverse level L or primary inputs. Next, the input values for reverse level i are determined by simulation of reverse levels i = L through reverse levels i = i-1. Parallel fault simulation of the reverse level i is now performed. The resulting set of potentially detected faults, W, are determined. Now, each potentially detected fault w in the set W is injected one at a time for fault simulation and simulated through the next reverse level. The fault w is dropped if undetected else it is simulated through the subsequent reverse level. This is repeated until the fault w is dropped as undetected or until it reaches the primary output and is considered detected. After all faults w in the set W have been injected and propagated, the next reverse level, i-1, is selected for fault simulation. This process is repeated for all reverse levels L to 1 and for all test vectors V.

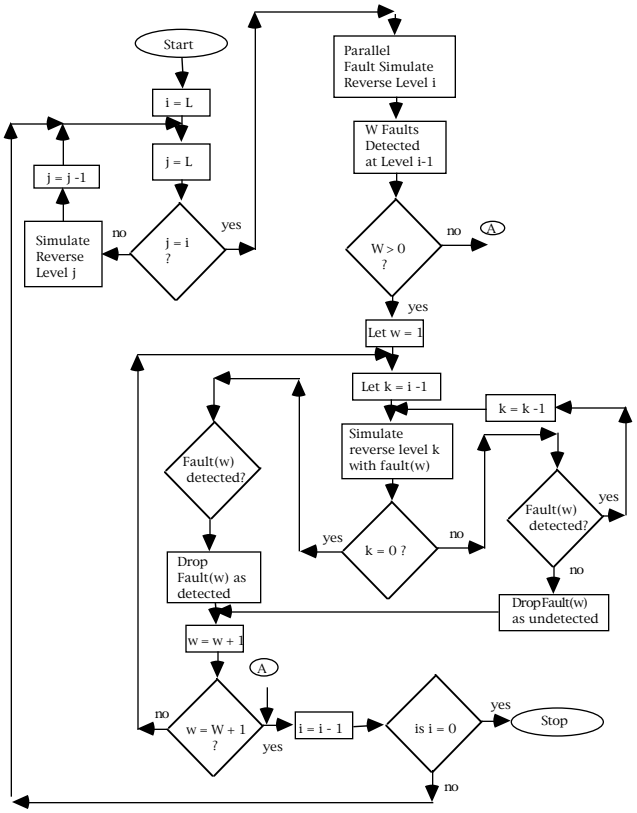

Figure 4.6 Parallel fault simulation flowchart.

An example compiled benchmark circuit, C17, is shown in Figure 4.7. One parallel switch (PS) performs parallel fault single pattern fault simulation for a single switch, while the nodes (NODESs) perform the fan-in operation at a node. Not shown in Figure 4.7 are the ground and Vdd inputs.

4.4 Verification Results

Fault simulation verification was performed on the seven smaller ISCAS85 [Brglez85] circuits. The input test vector sample sets used were obtained from the gate level test generation tool ATALANTA[1]. Fault simulation results for the switch level and the gate level are shown in Table 4.1. Results show, as expected, that gate level test vector sets detect less switch level faults. In this case results show that switch level fault coverage was about 40% less than the gate level fault coverage. This result, while not a surprise, does confirm the fact that switch fault simulation can be a better design verification tool. This is because a larger test set would be required to achieve a switch level fault coverage similar to the gate level fault coverage. Any possible switch level faults that are undetected for the gate level test set could be detected by the larger switch level test set.

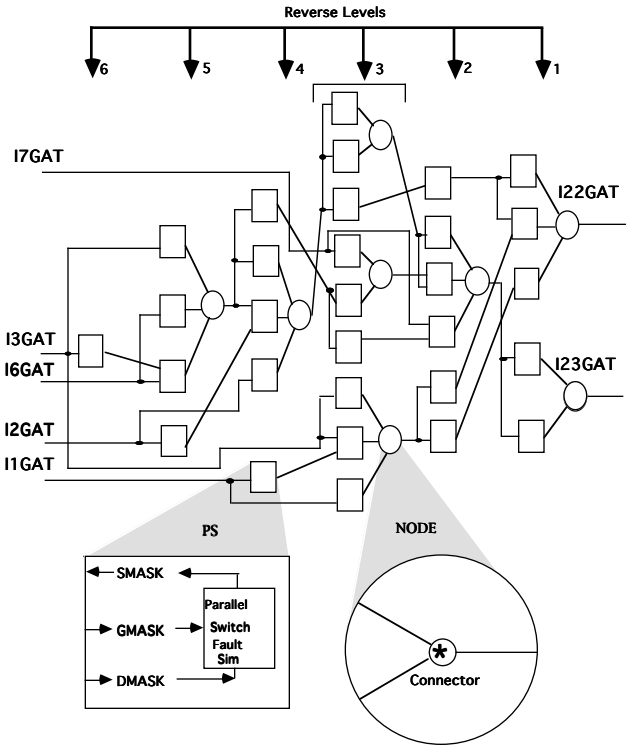

Figure 4.7 Compiled C17 reverse level ordering - PS's and NODES's.

Table 4.1 Fault simulation results.

CKT	#of switch	Gate detect Faults	Gate undet Faults	Gate Fault %Cover	Switch detect Faults	Switch undet Faults	Switch Xdet Faults	Switch Fault %Cover
c17	26	22	0	100.0	60	17	27	57.69
c432	728	523	1	99.81	1577	395	940	54.15
c499	1396	758	8	98.95	3065	518	2001	54.89
c880	1164	942	0	100.0	2712	633	1131	58.25
c1355	1769	1566	8	99.49	4061	918	2093	57.42
c1908	2058	1870	9	99.52	4740	1077	2415	57.58
c2670	2974	2630	117	95.74	6898	1266	3732	57.99

[1] Copyright Virginia Polytechnic Institute and State University 1991

Table 4.2 Fault simulation run times.

Circuit	Number of Switches	Number of test vectors	CPU (sec/ fault)
c17	26	7	.0477
c432	728	66	.056
c499	1396	54	.219
c880	1164	67	.065
c1355	1769	89	.105
c1908	2058	121	.249
c2670	2974	119	.254

Shown in Table 4.2 are these simulation time results for the benchmark circuits studied. The host computer used for simulation was a HP715. The VHDL simulator used was SYNOPSYS[Synop90]. The CPU time required for the switch level fault simulation was, as expected, much greater than that for the gate level. A quantitative comparison of these run time results with the run time results of the switch level simulators referenced is not possible because of the differing circuits, test sets and fault sets being reported. A qualitative comparison shows that the fault simulation times per fault are comparable to those reported in [Bryant85], while greater than those for the multi - level fault simulator reported in [Meyer93] . This comparison gives a strong case for using multi - level fault simulators as reported in [Meyer93] .

5. Summary

A novel VHDL switch level fault simulator has been presented. The simulator uses a switch level extension to PFSP fault simulation. The advantages of this technique and implementation are that it has the fault modeling accuracy of switch level simulators while keeping a straight forward compiled type fault simulator implementation. To preserved fault modeling accuracy, the PFSP switch level fault simulation *mask* operation is used to inject faults. While faults are injected at the individual switch, the *mask* operation is internal to the switch. The compiled type simulator is maintained because all switches (and injected faults) are compiled.

Acknowledgments

The authors gratefully acknowledge Synopsys Inc., Mountain View California USA, for providing the SYNOPSYS VHDL System Simulator. The authors would like to thank Krzysztof Kozminski of the Microelectronics Center of North Carolina for his assistance in providing the switch level descriptions of the ISCAS85 benchmark circuits. The authors would also like to thank Dong Ha of Virginia Polytechnic Institute and State University for his assistance in providing the gate level test vector generation tool ATALANTA.

Bibliography

[Barzi86] Z. Barzilai, D.K. Beece, L.M. Huisman, V.S. Iyengar, and G.M. Silberman, "SLS-A Fast Switch Level Simulator for Verification and Fault Coverage Analysis," IEEE 23rd Design Automation Conference, pp. 164 - 170, 1986.

[Bill91] W.D. Billowitch, VHDL Tech Notes, The VHDL Consulting Group, Bethlehem, Pa., June 1990.

[Bose82] A.K. Bose, P. Kozak, C-Y Lo, H.N. Nham, E. Pacas-Skewes, and K. Wu, "A Fault Simulator for MOS LSI Circuits," IEEE 19th Design Automation Conference Proceedings, pp. 400 - 409, 1982.

[Brglez85] F.Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," Proc. IEEE Intl Symp. on Circuits and Systems, Kyoto, pp. 663-698, June 1985.

[Bryant85] Randal E. Bryant, and Michael D. Schuster, "Performance Evaluation of FMOSSIM, a Concurrent Switch-Level Fault Simulator," IEEE 22nd Design Automation Conference Proceedings, pp. 715 - 719, 1985.

[Bryant87] R.E. Bryant, D. Beatty, K. Brace, K. Cho, and T. Sheffler, "COSMOS: A Compiled Simulator for MOS circuits," ACM IEEE 24th Design Automation Conference Proceedings, pp. 9 - 16, 1987.

[Fuji85] H. Fujiwara, "Logic Testing and Design for Testability," MIT Press, pp. 84-102, 1985.

[Hayes82] John P. Hayes, "A Fault Simulation Methodology for VLSI," IEEE 19th Design Automation Conference Proceedings, pp. 393 - 399, 1982.

[Hayes87] John P. Hayes, "An Introduction to Switch-Level Modeling," *IEEE Design and Test of Computers*, Vol. 6, No. 4, pp. 18-25, Aug. 1987.

[Kawai84] Masato Kawai, and John P.Hayes, "An Experimental MOS Fault Simulation Program CSASIM," IEEE 21st Design Automation Conference Proceedings, pp. 2 - 9, 1984.

[Lee91] T. Lee, and I. Haji, "A Switch-Level Matrix Approach to Transistor-Level Fault Simulation," Proc. ICCD, pp. 554-557, Nov. 1991.

[Light82] M.R. Lightner and G.D. Hachtel, "Implication Algorithms for MOS Switch Level Functional Macromodeling Implication and Testing," IEEE 19th Design Automation Conference Proceedings, 1982, pp. 691 - 698.

[Meyer93] W. Meyer, and Raul Camposano, "Fast Hierarchical Multi-Level Fault Simulation of Sequential Circuits witch Switch-Level Accuracy," ACM IEEE 30th Design Automation Conference Proceedings, pp 515 - 519, 1993.

[Shen85] John P. Shen, W. Maly, and F. Joel Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Design and Test of Computers*, Vol. 5, No. 4, pp. 13-26, Dec. 1985.

[Shih86] Hsi-Ching Shih, Joseph T. Rahmeh, and Jacob A. Abraham, "FAUST: An MOS Fault Simulator with Timing Information," *IEEE Transactions on Computer-Aided Design*, Vol. 5, No. 4, pp. 557-563, Oct. 1986.

[Synop90] Synopsys-The Synthesis Company, VHDL System Simulator Reference Manuel, 1990.