SCJ SOFT COMPUTING JOURNAL
ISSN: 2322-3707
University of Kashan

An Efficient Parallel Critical Path Tracing for Path Delay Fault Simulation

Hossein Sabaghian-Bidgoli[1,*], Ahmad Ehteram[2], Hossein Ghasvari[3], Majid Delshad[4] and Shaahin Hessabi[5]

[1] Department of Computer Engineering, University of Kashan, Kashan, Iran.
[2,3] Department of Electrical Engineering, Kashan Branch, Islamic Azad University, Kashan, Iran.
[4] Department of Electrical Engineering, Isfahan (Khorasgan) Branch, Islamic Azad University, Isfahan, Iran.
[5] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.

ABSTRACT. Delay fault simulation is the most general method that is used to assess the quality of generated test sets. Path delay fault is one of the most frequently used delay fault models. Path delay fault simulation is a time-consuming operation, especially for today's complex digital circuits. In this work, a novel critical path tracing algorithm is proposed for parallel path delay fault simulation. The obtained outcomes denote 489 times average speedup compared with the traditional path tracing, as well as 186 times average speed-up in comparison with the latest reported results of previous studies.

KEYWORDS: Path delay fault, Fault simulation, Critical Path Tracing, Robust path, Non-robust path.

1 INTRODUCTION

Recent advances in digital circuit manufacturing, along with the increasing complexity of demanded products, have led to large scale integration and then more likelihood of faults, thus increasing the importance of testing [1]. Delay faults, which describe a type of permanent faults are tested using test pairs. Test generation for a type of fault requires a suitable fault model. Path delay fault is one of the most popular delay fault models in which a delay fault is associated with a path that connects an input to an output. Since the number of paths exponentially increases with the size of the circuit, test generation and fault simulation for large circuits is a serious challenge [2].

Studies on reducing the time of path delay fault simulation can be generally divided into three categories. The first category [3-4], called enumerative methods, examines each path individually for detection by the given tests. The main problem with these methods is their long execution time. The second category, called non-enumerative methods, consists of methods that avoid counting individual paths [5-7]. Although these algorithms are fast, they are not able to obtain the exact amount of fault coverage. The third category considers all paths to maintain accuracy and uses GPUs as accelerators to increase speed [8-9]. The need for special hardware such as GPU is a disadvantage of these methods. In other words, these algorithms cannot run on every hardware.

We propose a very fast enumerative path delay fault simulation algorithm which increases the speed while preserving its accuracy [10]. The proposed algorithm does not require special hardware and works on any system with a general processor. This paper is an extended version of our previous work [10]. In contrast to that, in the present work, the effect of each different technique on increasing the speed has been studied independently and also in combination, and represented graphically. The classification of critical, sensitive and robust paths are more clearly stated, their CPT formulas are reviewed and revised, the concept of non-sensitive paths is added, and the relationship between non-robust and non-sensitive paths with the three main categories is graphically expressed. The rest of this article is organized as follows. The proposed method is presented in Section 2. In Section 3, the experimental results are reported and discussed, and Section 4 summarizes the proposed method and concludes the paper.

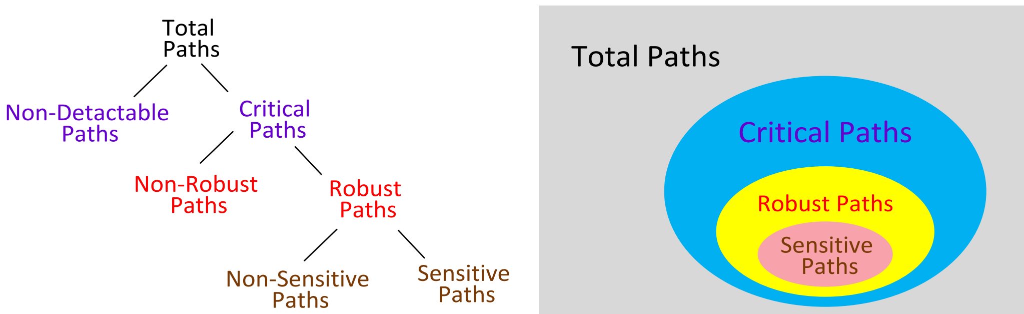


Figure 1: Graphical representation of different path sets

2 PROPOSED METHOD

This method simultaneously uses several different techniques to reduce the overall simulation time. Three novel techniques are as follows:

- Simplifying the propagation conditions check by combining the robust and non-robust paths (called critical paths) and considering the propagation condition of their union which are simpler conditions. (See Figure 1)
- Critical path tracing expansion for path delay fault simulation separately for critical, strong and sensitive paths. Provided formulas for three types of paths are shown in Table 1.
- Creating an array checklist based on the path index with the aim of eliminating the search

Table 1. Proposed CPT Formulas for Critical, Robust and Sensitive paths

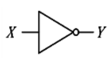
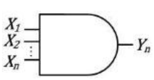
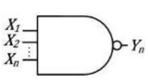
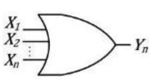
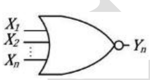
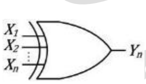
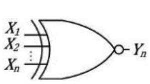
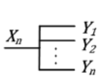
Gate	Function	Logic Value	Criticality of Critical path (C_{cp}), Criticality of sensitive path (C_{sp}) and Criticality of robust path (C_{rp})
	$Y = NOT(X)$	$V_1(Y) = \overline{V_1}(X)$ $V_2(Y) = \overline{V_2(X)}$	$C_{cp}(X) = C_{cp}(Y)$ $C_{sp}(X) = C_{sp}(Y)$ $C_{rp}(X) = C_{rp}(Y)$
	$Y_n = AND_{i=1}^{n}(X_i)$	$V_1(Y_n) = \prod_{i=1}^{n} V_1(X_i)$ $V_2(Y_n) = \prod_{i=1}^{n} V_2(X_i)$	$C_{cp}(X_k) = C_{cp}(Y_n) \prod_{i=1, i\neq k}^{n} V_2(X_i)$ $C_{sp}(X_k) = C_{sp}(Y_n) \left[\prod_{i=1, i\neq k}^{n} V_1(X_i)\right]\left[\prod_{i=1, i\neq k}^{n} V_2(X_i)\right]$
	$Y_n = NAND_{i=1}^{n}(X_i)$	$V_1(Y_n) = \overline{\prod_{i=1}^{n} V_1(X_i)}$ $V_2(Y_n) = \overline{\prod_{i=1}^{n} V_2(X_i)}$	$C_{rp}(X_k) = C_{rp}(Y_n) \left[\prod_{i=1, i\neq k}^{n} V_2(X_i)\right]\left(\left[\prod_{i=1, i\neq k}^{n} V_1(X_i)\right] V_1(X_k)\overline{V_2}(X_k) + \overline{V_1}(X_k)V_2(X_k)\right)$ $k = 1, 2, \dots, n$
	$Y_n = OR_{i=1}^{n}(X_i)$	$V_1(Y_n) = \overline{\prod_{i=1}^{n} \overline{V_1}(X_i)}$ $V_2(Y_n) = \overline{\prod_{i=1}^{n} \overline{V_2}(X_i)}$	$C_{cp}(X_k) = C_{cp}(Y_n) \prod_{i=1, i\neq k}^{n} [1 - V_2(X_i)]$ $C_{sp}(X_k) = C_{sp}(Y_n) \left[\prod_{i=1, i\neq k}^{n} \overline{V_1}(X_i)\right]\left[\prod_{i=1, i\neq k}^{n} \overline{V_2}(X_i)\right]$
	$Y_n = NOR_{i=1}^{n}(X_i)$	$V_1(Y_n) = \prod_{i=1}^{n} \overline{V_1}(X_i)$ $V_2(Y_n) = \prod_{i=1}^{n} \overline{V_2}(X_i)$	$C_{rp}(X_k) = C_{rp}(Y_n) \left[\prod_{i=1, i\neq k}^{n} \overline{V_2}(X_i)\right]\left(\left[\prod_{i=1, i\neq k}^{n} \overline{V_1}(X_i)\right] \overline{V_1}(X_k)V_2(X_k) + V_1(X_k)\overline{V_2}(X_k)\right)$ $k = 1, 2, \dots, n$
	$Y_n = XOR_{i=1}^{n}(X_i)$	$Y_{n-1} = XOR_{i=1}^{n-1}(X_i) \oplus X_n$ $V_1(Y_n) = V_1(Y_{n-1}) \oplus V_1(X_n)$ $V_2(Y_n) = V_2(Y_{n-1}) \oplus V_2(X_n)$	$C_{cp}(X_k) = C_{cp}(Y_n)$ $C_{sp}(X_k) = C_{sp}(Y_n) \left(\left[\prod_{i=1, i\neq k}^{n} V_1(X_i)\right]\left[\prod_{i=1, i\neq k}^{n} V_2(X_i)\right] + \left[\prod_{i=1, i\neq k}^{n} \overline{V_1}(X_i)\right]\left[\prod_{i=1, i\neq k}^{n} \overline{V_2}(X_i)\right]\right)$
	$Y_n = XNOR_{i=1}^{n}(X_i)$	$Z_n = XOR_{i=1}^{n}(X_i)$ $Z_{n-1} = XOR_{i=1}^{n-1}(X_i) \oplus X_n$ $V_1(Y_n) = \overline{V_1}(Z_n)$ $V_2(Y_n) = \overline{V_2}(Z_n)$	$C_{rp}(X_k) = C_{rp}(Y_n) \left(\left[\prod_{i=1, i\neq k}^{n} V_1(X_i)\right]\left[\prod_{i=1, i\neq k}^{n} V_2(X_i)\right] + \left[\prod_{i=1, i\neq k}^{n} \overline{V_1}(X_i)\right]\left[\prod_{i=1, i\neq k}^{n} \overline{V_2}(X_i)\right]\right)$ $k = 1, 2, \dots, n$
	n branch Fanout $Y_i = X_n$ $i = 1, 2, \dots, n$	$V_1(Y_i) = V_1(X_n)$ $V_2(Y_i) = V_2(X_n)$ $i = 1, 2, \dots, n$	$C_{cp}(X_n) = 1 - \prod_{i=1}^{n} \overline{C_{cp}}(Y_i)$ $C_{sp}(X_n) = 1 - \prod_{i=1}^{n} \overline{C_{sp}}(Y_i)$ $C_{rp}(X_n) = 1 - \prod_{i=1}^{n} \overline{C_{rp}}(Y_i)$

Table 2. Comparing the PDF simulation times of different path sets for 10000 random tests

		Critical Path		Robust Path		Sensitive Path		Non-Robust Path		Non-Sensitive Path	
Benchs	**#TotalPaths**	**#Paths**	**Time(s)**	**#Paths**	**Time(s)**	**#Paths**	**Time(s)**	**#Paths**	**Time(s)**	**#Paths**	**Time(s)**
b11_C	**21144**	5856	0.094	2449	0.047	1475	0.046	3407	0.141	974	0.093
b12_C	**25788**	9409	0.14	5502	0.094	2532	0.078	3907	0.234	2970	0.172
b13_C	**1398**	1130	0.047	941	0.031	834	0.031	189	0.078	107	0.062
b14_C	**186784982**	452156	0.875	58319	0.531	9046	0.484	393837	1.406	49273	1.015
b15_C	$2^{36}<\#<2^{37}$	259739	1.094	29917	0.844	8378	0.781	229822	1.938	21539	1.625
b17_C	$2^{40}<\#<2^{41}$	749766	4.11	95435	2.688	25956	2.406	654331	6.798	69479	5.094
c1355	**8346432**	327454	0.156	2595	0.047	121	0.031	324859	0.203	2474	0.078
c1908	**1458114**	37896	0.125	3721	0.078	2223	0.063	34175	0.203	1498	0.141
c2670	**1359920**	52228	0.235	7875	0.109	4101	0.093	44353	0.344	3774	0.202
c3540	**57353342**	364710	0.421	20227	0.125	387	0.062	344483	0.546	19840	0.187
c432	**167852**	9707	0.031	1860	0.016	406	0.016	7847	0.047	1454	0.032
c499	**18880**	10812	0.047	2246	0.016	101	0.016	8566	0.063	2145	0.032
c5315	**2682610**	152265	0.437	15213	0.203	7995	0.157	137052	0.64	7218	0.36
c7552	**17284**	159199	0.782	24687	0.265	11122	0.218	134512	1.047	13565	0.483
s15850	**329476092**	283018	1.437	14170	0.812	8132	0.734	268848	2.249	6038	1.546
s38417	**2783158**	202382	2.641	63664	1.985	32172	1.703	138718	4.626	31492	3.688
s38584	**2161446**	87486	2.469	40981	1.938	25989	1.797	46505	4.407	14992	3.735

operation while merging the list of newly detected paths in the list of detected paths so far. The combination of the three proposed techniques with the two conventional techniques, namely 32-bit parallelism and path indexing, leads to significant speed-up, which is reported in the next section

3 EXPERIMENTAL RESULTS

The proposed algorithm was implemented in C++ and ran on a system with a 3.6 GHz Core i7 with 16-GB RAM. The experiments were performed on a selected set of ISCAS'85, ISCAS'89, and ITC'99 benchmarks. The simulation results of different sets of path delay faults are shown in Table 2. The results of critical, robust and sensitive paths are directly obtained by using path delay fault simulation while the results of the non-robust and non-sensitive paths are obtained using the results of the three first path sets.

In addition, the results of non-robust paths are compared with a number of recent reports from similar studies in Table 3. The results show the effectiveness of the proposed techniques and the significant improvement in runtime compared to the work of others.

Figure 2 depicts the contribution of three techniques, including path indexing, critical path tracing, and bit parallelism on speed-up. One of the most interesting points in this diagram is the tremendous effect of combining two techniques, parallelism (32-bit) and critical path tracing (CPT) over traditional path tracing (TPT). While the effectiveness of each one individually gives 2.93 and 2.16 speed-ups, respectively, the combination of them results in 28.63 speed-up.

Table 3. Comparing the execution time (sec) of approximate non-robust path delay fault Simulation using 10000 random tests with the state-of-the-art methods

Circuit	**Total path**	**[3]**	**[4]**	**Minimum**	**Our method**	**Speedup**
c432	83926	**17.53**	42	17.53	0.047	372.98
c499	9440	**19.22**	-	19.22	0.063	305.08
c880	8642	29.81	**26**	26	0.140	185.71
c1355	4173216	215.04	**79**	79	0.203	389.16
c1908	729057	**92.70**	94	92.70	0.203	456.65
c2670	679960	429.05	**30**	30	0.344	87.21
c3540	28676671	3161.8	**166**	166	0.546	304.03
c5315	1341305	487.76	**74**	74	0.640	115.63
c7552	726494	628.36	**98**	98	1.047	93.60
			Average:	66.94	0.359	186.46

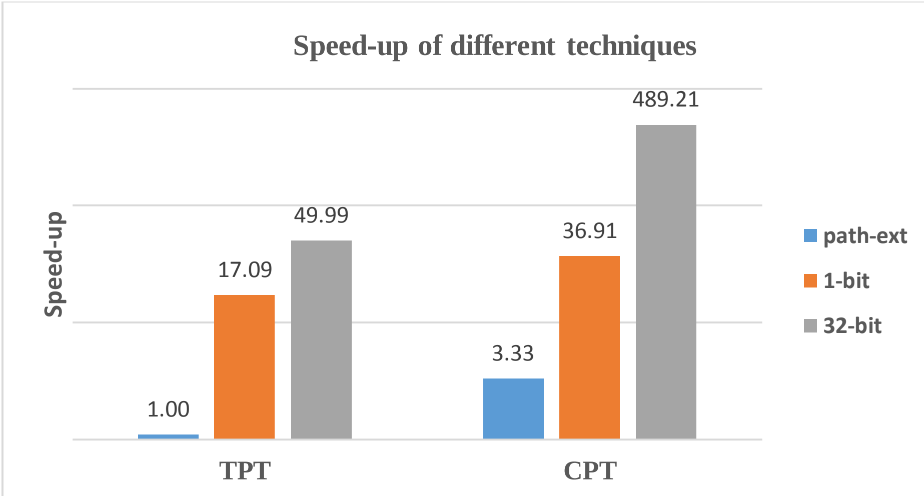


Figure 2: Comparing Speed-up of different techniques

4 CONCLUSION

A very fast CPT-based method was proposed for path delay fault simulation. This method eliminates many undetectable paths at the earlier step (backward tracing). Thus, it has a great effect on reducing computations and increasing the simulation speed, especially when combined with bit parallelism technique.

REFERENCES

1. A. Ehteram, H. Sabaghian-Bidgoli, H. Ghasvari and S. Hessabi, "A Simple and Fast Solution for Fault Simulation Using Approximate Parallel Critical Path Tracing," in Canadian Journal of Electrical and Computer Engineering, vol. 43, no. 2, pp. 100-110, Spring 2020, doi: 10.1109/CJECE.2019.2950280
2. A. Krstic and K.-T. Cheng, Delay Fault Testing for VLSI Circuits, US, Springer, 1998, doi: 10.1007/978-1-4615-5597-1
3. A. K. Majhi, J. Jacob and L. M. Patnaik, "A Novel Path Delay Fault Simulator using Binary Logic", VLSI Design, vol. 4, pp. 167-179, 1996, doi: 10.1155/1996/25839
4. P. Manikandan, B. B. Larsen and E. J. Aas, "An Enhanced Path Delay Fault Simulator for Combinational Circuits," 14th Euromicro Conference on Digital System Design, Oulu, Finland, 2011, pp. 375-381, doi: 10.1109/DSD.2011.52
5. I. Pomeranz and S. M. Reddy, "An efficient non-enumerative method to estimate path delay fault coverage," IEEE/ACM International Conference on Computer-Aided Design, Santa Clara, CA, USA, 1992, pp. 560-567, doi: 10.1109/ICCAD.1992.279312
6. I. Pomeranz and S. M. Reddy, "An efficient nonenumerative method to estimate the path delay fault coverage in combinational circuits," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 2, pp. 240-250, 1994, doi: 10.1109/43.259947
7. K. Heragu, V. D. Agrawal and M. L. Bushnell, "Statistical methods for delay fault coverage analysis," Proceedings of the 8th International Conference on VLSI Design, New Delhi, India, 1995, pp. 166-170, doi: 10.1109/ICVD.1995.512098
8. Y. Ali, Y. Yamato, T. Yoneda, K. Hatayama and M. Inoue, "Parallel Path Delay Fault Simulation for Multi/Many-Core Processors with SIMD Units," IEEE 23rd Asian Test Symposium, Hangzhou, China, 2014, pp. 292-297, doi: 10.1109/ATS.2014.61
9. E. Schneider, M. A. Kochte, S. Holst, X. Wen, and H.-J. Wunderlich, "GPU-accelerated simulation of small delay faults," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 5, pp. 829–841, 2017, doi: 10.7873/DATE.2015.0077
10. A. Ehteram, H. Sabaghian-Bidgoli, H. Ghasvari and S. Hessabi, "A very fast algorithm for path delay fault simulation of digital circuit based on parallel critical path tracing", Soft Computing Journal (SCJ), vol. 9, no. 1, pp. 124–145, 2020, doi: 10.22052/SCJ.2021.111571