Fault Simulation and Random Test Generation for Speed-Independent Circuits

Feng Shi
Electrical Engineering Dept.
Yale University
New Haven, CT 06520, USA

Yiorgos Makris
Electrical Engineering Dept.
Yale University
New Haven, CT 06520, USA

ABSTRACT

We develop a fault simulator for input stuck-at faults in Speed-Independent circuits by extending Eichelberger's method. In order to achieve higher accuracy, a 13-valued algebra is adopted, the relative order of causal signal transitions is maintained, and time frames are unfolded in a careful manner. Based on this simulator, we propose a random test generation algorithm which reduces the probability that the circuit finds itself in non-deterministic states and helps it recover when this happens. Experimental results show that the combination of the two techniques achieves an average improvement of 18% in fault coverage.

Categories and Subject Descriptors

B.7.3 [**Integrated Circuits**]: Reliability and Testing

General Terms

Algorithms, Reliability, Verification

Keywords

Asynchronous Circuits, Speed-Independent Circuits, Fault Simulation, Random Test Pattern Generation

1. INTRODUCTION

Interest in asynchronous circuits has recently been revived, not only because of their potential for high performance, low power and design reusability, but also because of the difficulties associated with traditional synchronous design such as clock skew. As an alterative to synchronous circuits, asynchronous circuits have already started to demonstrate their potential, even in many commercial products.

However, one of the main obstacles in the widespread development of asynchronous circuits is the difficulty in testing them. Without a global clock for synchronization, asynchronous circuits operate in a rather independent manner, which is sensitive to race conditions and hazards. In addition, it is much harder to control and observe internal nodes in asynchronous circuits. Moreover, there are several classes of asynchronous circuits such as Delay-Insensitive, Speed-Independent, Timed circuits, and so on. Fault simulation and test generation methods for one class of circuits might not work for other classes, which makes the problem even more complicated.

In recent years, numerous efforts have been devoted to fault modelling, fault simulation and test generation for asynchronous circuits. Several researchers [1, 2, 11] studied the problem of testing asynchronous circuits using available commercial testers for synchronous circuits. In order to avoid uncertainty, fault effects have to be observed when the circuit is in a stable state. Since many asynchronous circuits operate in fundamental mode, test vectors can only be applied after the circuit has stabilized. Hazard/race analysis is critical in simulation and test generation of asynchronous circuits. Eichelberger [7] first used ternary logic simulation for detecting hazards in both combinational and sequential logic. He also proposed a method for simulation of sequential circuits. Subsequently, multi-valued logic has been used in different contexts. For instance, Chakraborty, Bushnell and Agrawal [4] used the 13-valued algebra for delay fault test generation. Fsimac, a gate-level fault simulator for stuck-at and gate-delay faults in extended burst mode machines was developed in [12], using min-max timing analysis [3] and 13-valued logic. Fsimac assumes fundamental mode of operation and simulates both the good circuit and the bad circuit in a time-unfolding manner until the primary outputs and state signals stabilize.

The Fsimac assumption of fundamental operation mode is valid only for Huffman circuits. Speed-Independent circuits, however, do not impose any restrictions on the order that inputs, outputs, and state signals change, except that they must behave according to the protocol [10]. Hence, simple time-frame unfolding simulation methods for Huffman circuits might be invalid for these circuits, since they ignore the additional race conditions and hazards imposed by changes of state signals before the circuit has stabilized. Eichelberger's method [7] for simulation of sequential circuits can be adapted for simulation of Speed-Independent circuits as in [11], but it is unacceptably conservative to be used for a fault simulator. In many cases, it fails to determine the actual circuit state and simply reports an unknown state.

In this paper we propose a fault simulation algorithm for Speed-Independent circuits by extending Eichelberger's method. We first describe our algorithm in section 2. Then, a random test generation method based on the proposed fault-simulator is given in section 3. Experimental results are provided in section 4.

2. PROPOSED SIMULATION METHOD

Eichelberger developed an algorithm for determining the next stable state of an asynchronous circuit in [7], which can be adapted and used for simulation of Speed-Independent circuits. His method used $\frac{1}{2}$ to denote an unknown state, which we typically denote by

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GLSVLSI'04, April 26–28, 2004, Boston, Massachusetts, USA.
Copyright 2004 ACM 1-58113-853-9/04/0004 ...$5.00.

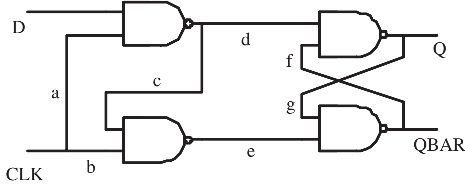

Figure 1: Gate-Level Schematic of a D-Latch

X. Suppose each feedback line is cut with one end denoted as a pseudo primary input (PPI) and the other as a pseudo primary output (PPO). The algorithm consists of two procedures. In procedure A, all changing state signals are determined by setting changing primary inputs (PIs) to X and all other PIs and PPIs as originally specified, and then evaluating the PPOs. If there are any PPOs equal to X, the corresponding PPIs are changed to X and the process is repeated until no additional changes occur in PPOs. Then, Procedure B takes over to determine which value each state signal stabilizes to. With the changing PIs equal to their new values and all other PIs and PPIs equal to their values at the end of Procedure A, PPOs and primary outputs (POs) are evaluated. If one or more PPOs change from X to 1 or 0, the corresponding PPI is changed and the process is repeated until no additional changes occur in PPOs.

As observed in [11], Eichelberger's method is too conservative to be used as a fault simulation algorithm for Speed-Independent circuits. First, ternary logic is not able to discriminate rising/falling transitions from unknown states or glitches. Second, like most other multi-valued algebras, it cannot express the order of signal transitions, which leads to false indication of hazards. Third, Eichelberger's method sets the changing state signals to X in Procedure A and then assigns deterministic values to some of them in Procedure B. Therefore, only those state changes forced directly or indirectly by PIs and/or stable state signals are correctly simulated, while other state changes are reported as undetermined, although in many cases they are not. As a result, Eichelberger's method fails to determine the next state of a Speed-Independent circuit in a considerable number of cases, which makes it unsuitable for fault simulation.

In order to overcome these problems, we extend Eichelberger's method in several ways. First, we use the 13-valued algebra, as in [3, 4, 12], to represent signal transitions more accurately. In addition, this method is compact since it avoids unnecessary event proliferations by abstracting the details of multi-transition waveforms. The extension of gate functions from the 3-valued logic to the 13-valued logic is not difficult and is detailed in the above references. Another reason for using 13-valued logic is that it facilitates the expression of functions of some complicated gates, such as Muller C-elements, latches and complex domino gates, which are widely used in asynchronous circuits.

Second, the relative order of signal transitions is maintained by keeping a simple time stamp during gate evaluation. Since in Speed-Independent circuits we normally assume the *unbounded gate delay* model [9], (i.e. delay elements are attached only to gate outputs and the delay magnitude is positive and finite but unknown,) and the *pure delay* type, (i.e. waveforms are shifted in time and do not change shape,) the relative order of causal signal transitions is necessary for correct simulation in many cases. For example, suppose that the initial values of PIs of the D-Latch in figure 1 are $CLK = 1$ and $D = 1$, the values of the internal nodes are $abcdefg = 1100101$, and the values of the PPOs are $Q = 1$ and $QBAR = 0$. Now CLK falls to 0 and D doesn't change, or in 13-valued logic, $CLK =< 1,\downarrow,0 >$ and $D =< 1,1,1 >$. Therefore, $b =< 1,\downarrow,0 >$, $c =< 0,\uparrow,1 >$ and $e =< 1,X,1 >$ by gate evaluation, and a glitch on e is reported. But in fact the circuit assumes that a and b fall simultaneously, and since the gate delay for c is positive, c rises after a falls, hence after b falls, so there is no glitch on e. Since $d = c =< 0,\uparrow,1 >$, it's easy to find that $Q =< 1,1,1 >$ and $QBAR =< 0,0,0 >$. If we simulate the circuit under the above stimulus using Eichelberger's method, Q and $QBAR$ are set to X at the end of Procedure A, and they are not recovered in Procedure B, so the simulation gives an undetermined result. In the proposed algorithm, one simple time stamp is maintained for each transition that has a causal order with other transitions. Therefore, we can correctly evaluate when input transitions have a relative order, and we can deal with isochronic forks.

Third, our method carefully unfolds time frames. Unlike in Procedure A of Eichelberger's algorithm, which treats transitions and hazards in the same way, if there is any hazard but no transition detected on any PPO after evaluation, the corresponding PPI is set to the undefined value and the process is repeated until no more hazards are detected. Then, if there is any transition detected on any PPO, the corresponding PPI is set to the transition, the circuit is reevaluated and any hazards are handled as previously. This process is repeated until no more hazards or transitions occur on PPOs or until the number of iterations exceeds the pre-specified maximum limit. It's necessary to set a maximum number of iterations to break the loop if the circuit oscillates, in which case the first terminating condition will never be satisfied. After this step, a procedure similar to Procedure B of Eichelberger's algorithm takes place to determine the stabilizing values on some POs and state signals.

Moreover, the proposed method collapses faults before simulation. We adopt the definitions and notation in [5], and refer to the concepts of fault equivalence/dominance in a single gate as *g*-equivalence/dominace, in a combinational circuit as *c*-equivalence/dominace. It is obvious that the equivalence relationship in a single gate remains valid in a Speed-Independent circuit. Unfortunately, a *c*-equivalent pair of faults might not be equivalent in a Speed-Independent circuit, since the circuit under each fault might have different hazard conditions that lead to different nondeterministic states. Therefore, a test for one fault in a *c*-equivalent pair might be invalid for the other. For the same reason, as well as due to self-hiding and delayed reconvergence [5], a *c*-dominant and *c*-dominated pair of faults might not be a dominant and dominated pair in a Speed-Independent circuit. In this work we only collapse conservatively, i.e. only *g*-equivalent faults.

3. RANDOM TPG METHOD

In Speed-Independent circuits, random test pattern generation remains an efficient and important method. However, the special properties of Speed-Independent circuits may degrade its efficiency. A Speed-Independent circuit is guaranteed to work only on input stimuli allowed by the specification. Since random test generation does not consider the specification of the circuit under test, invalid test patterns are very likely to be generated. Moreover, once it is trapped into an unknown state, the circuit normally will have difficulty in getting out. Therefore, long yet meaningless sequences of test patterns are typically generated by random test generation algorithms.

We overcome this problem by resetting the circuit with a probability that is dynamically adjusted. The proposed random test generation algorithm can be aware of invalid circuit states by checking the values on the feedback lines. Moreover, the algorithm decides whether or not to reset the circuit based on a probability that is in proportion to the number of unstable feedback lines. A larger number of unstable feedback lines normally means a greater need for initializing the circuit, although it might not be necessary to reset

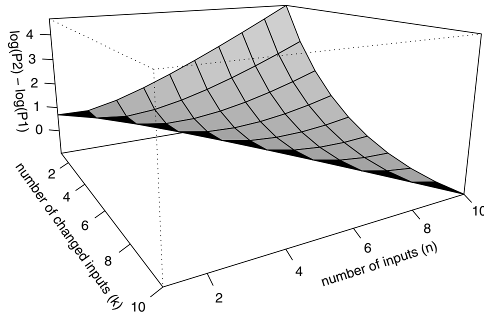

Figure 2: Probabilities of MIC and SIC Vectors

the circuit since the fault may still be detected through another stable output. An unstable value on any feedback line always means the circuit needs to be initialized. By monitoring the circuit states and by probabilistically initializing, the circuit can recover from undetermined states.

Although a Speed-Independent circuit responds correctly only to the specified interface behavior, it doesn't put any restrictions on the speed or order of the input changes. Therefore, if an input vector with multiple-input changes (MIC) is valid for a Speed-Independent circuit and the circuit goes to a certain state after applying the vector, it must go to the same state after applying a sequence of vectors which is obtained by arbitrarily breaking down the MIC vector into several vectors, each of which only allows single-input change (SIC). For example, if a circuit transfers to state B from state A under the input stimulus 0110 after 0000, and it's obvious that the MIC sequence 0000 → 0110 can be broken down into SIC sequences 0000 → 0100 → 0110 or 0000 → 0010 → 0110, the circuit must also go to state B from A after applying any of the above SIC sequences. From a test generation perspective, if a fault in a Speed-Independent circuit is detected under a test sequence which includes MIC vectors, the fault must also be detected under the SIC test sequence which is obtained by arbitrarily breaking down each of the MIC vectors into a sequence of SIC vectors. So our random test generation algorithm only allows SIC test sequences. One reason for this restriction is that a MIC vector, particularly with a large number of input changes, is more hazardous and more likely to lead the circuit into an undetermined state. By applying only relatively safer SIC test vectors the circuit has fewer opportunities to encounter hazard conditions. A disadvantage of this technique is that it may lead to longer test sequences because it allows only one-bit change at a time. However, these longer test sequences can be compacted in a later phase.

It is interesting to look at the difference between the probability of generating a MIC vector in the normal random test generation algorithm and that of generating the corresponding SIC sequence, called a MIC vector alternative, in the proposed algorithm. We assume that the number of inputs is n, and each input has a uniform and independent probability of $\frac{1}{2}$ of being flipped in the normal algorithm, and every input has equal opportunity to be chosen to change in the algorithm only allowing SIC. Therefore, the probability of generating a MIC vector with k inputs changed in the normal algorithm is

$$P_1 = (\frac{1}{2})^n \tag{1}$$

while the probability of generating its SIC sequence alternative in the proposed algorithm is

$$P_2 = k! \times (\frac{1}{n})^k \tag{2}$$

since any possible alternative has k vectors hence the probability of generating it is $(\frac{1}{n})^k$, and there is a total of $k!$ such possible alternatives. By plotting the difference $(\lg P_2 - \lg P_1)$ between P_1 and P_2 in figure 2, we can find out that P_2 is larger than P_1 when k is much smaller than n, but P_2 is smaller than P_1 when k becomes large and comparable to n. That is to say, the proposed method prefers to generate a MIC vector alternative with a small number of input changes than that with a large number of input changes. Fortunately, this is in line with the fact that, typically, only a limited number of inputs change at each time in an asynchronous circuit. Therefore the proposed method has a high probability of generating patterns that are likely to be valid and a low probability of generating patterns that are unlikely to be valid.

4. EXPERIMENTAL RESULTS

We developed a fault simulator for Speed-Independent circuits based on HOPE [8]. The input circuit netlist is in ISCAS89 format, and the stuck-at fault list can be defined in a input file or generated automatically by the tool. In the latter case, all stuck-at faults on gate inputs and outputs are injected and g-equivalent faults are collapsed. The proposed simulation algorithm for Speed-Independent circuits is applied for each test vector, first on the good circuit, then on faulty circuits with a single stuck-at fault injected. Output values are then compared to identify detected faults. The proposed random test generation algorithm was implemented as described in section 3. It terminates when there is no new fault detected for 20 consecutive test vectors. The generated patterns are guaranteed to detect single stuck-at faults assuming any possible combination of gate delays.

We experimented with the proposed algorithms on a set of Speed-Independent circuits synthesized by Petrify [6]. Complex gates like Muller C-elements are replaced by their gate-level implementations since the simulator cannot yet handle them directly. Note that a Speed-Independent circuit might not remain Speed-Independent any longer after such replacements. In each of the benchmark circuits, a reset input port is assumed to be connected to every memory element to appropriately initialize the circuit.

In order to demonstrate the efficiency of the proposed fault simulation and random test generation algorithms, we also implemented Eichelberger's simulation method and a normal random test generation algorithm for the purpose of comparison. We implemented Eichelberger's method as in [7] except that 13-valued algebra is adopted during the simulation. The normal random test generation algorithm assumes that each input has an independent probability of $\frac{1}{2}$ to change. For each setting and each circuit we repeated the experiment for 100 times and we report the average.

Table 1 illustrates the experimental results of the proposed fault simulation and random test generation methods for Speed-Independent circuits. The total number of stuck-at faults before collapsing, after collapsing, and the collapsing rate for each circuit are listed in the second, third, and fourth columns, respectively. Note that the number of total faults for each circuit is normally larger than that in [11], because we substituted complex gates with gate-level implementations, hence there are more gates and more possible faults in each circuit. Unfortunately some of the new faults have been proven to be redundant. Although we only collapsed g-equivalent faults, the fault collapsing rate is still considerable, averaging at 42.1%. The fifth and sixth columns present the average number of detected faults and the average fault coverage for each circuit. An average fault coverage of 76.5% is achieved across all circuits. In order to present the detailed behavior of the random test generation algorithm, we also list the maximum fault coverage and the standard deviation in the 100 experiments for each circuit in the seventh and eighth columns, respectively.

Circuit Name	No. of Faults	No. of faults After collapsing	Collapsing Rate	Average No. of Detected faults	Average fault Coverage	Max fault Coverage	Standard deviation Of fault Coverage
alloc-outbound	138	82	41%	75.7	92%	95%	0.038
chu133	78	44	44%	42.5	97%	98%	0.033
chu150	98	60	39%	48.8	82%	82%	0.010
converta	106	67	37%	30.6	46%	52%	0.074
dff	86	50	42%	39.7	79%	82%	0.099
hazard	86	52	40%	44.6	86%	86%	0.015
master-read	306	174	43%	80.5	46%	56%	0.072
mp-forward-pkt	116	68	41%	64.6	95%	97%	0.029
mr1	316	184	42%	73.8	40%	60%	0.191
nak-pa	166	94	43%	85.5	91%	94%	0.060
nowick	76	40	47%	39.2	98%	100%	0.042
ram-read-sbuf	170	90	47%	87.2	89%	97%	0.077
rcv-setup	62	36	42%	33.3	93%	94%	0.034
rpdft	88	47	47%	43.0	92%	96%	0.039
sbuf-ram-write	214	128	40%	99.9	78%	95%	0.129
sbuf-send-ctl	190	110	42%	53.7	49%	62%	0.131
seq4	214	130	39%	60.9	47%	67%	0.097
Average	147.6	85.6	42.1%	59.0	76.5%	83.1%	0.069

Table 1: Experimental Results

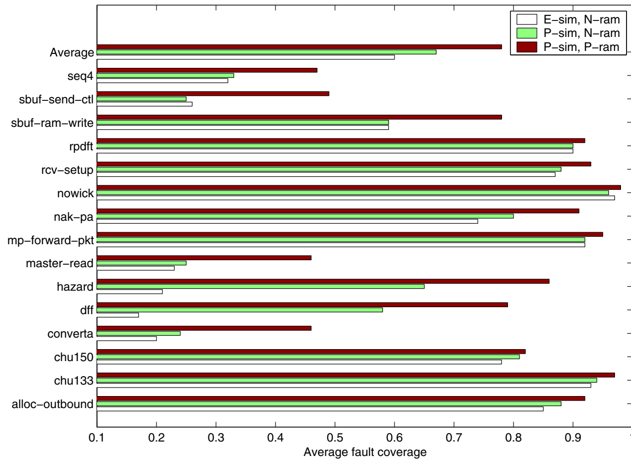

Figure 3: Fault Coverage Improvement

We compare the proposed fault simulation and random test generation methods to Eichelberger's method and a normal random method. Figure 3 presents the experimental results of three combinations of the methods for each circuit. The three combinations of the methods are Eichelberger's method and the normal random method (E-sim, N-ram), the proposed simulation method and the normal random method (P-sim, N-ram), and the proposed simulation and random methods (P-sim, P-ram) respectively. On average, the proposed simulation method increases the fault coverage by 7%, and the proposed random test generation algorithm achieves additional 11%.

5. CONCLUSION

Speed-Independent circuit simulation requires consideration of hazard conditions caused by changes of feedback lines before the circuit stabilizes. Similarly, Speed-Independent circuit testing necessitates that faults be detected when the circuit is in a stable state. Towards this end, we proposed a fault simulation algorithm for input stuck-at faults in Speed-Independent circuits by extending Eichelberger's method [7]. In addition to adopting a 13-valued algebra, we also maintain the relative order of causal signal transitions in the circuit, and we carefully unfold time frames to achieve better accuracy at a low computational cost. We discussed fault collapsing in Speed-Independent circuits, and we proposed a random test generation method that assists Speed-Independent circuits in exiting from nondeterministic states by resetting with a dynamically adjustable probability. Moreover, we reduced the probability that circuits enter nondeterministic states by only allowing test vectors with SIC. Experimental results demonstrate a fault collapsing rate of 42%. The proposed fault simulation algorithm improves fault coverage by 7%, while random test generation yields an additional 11%.

6. REFERENCES

[1] S. Banerjee, S. T. Chakradhar, and R. K. Roy. Synchronous test generation model for asynchronous circuits. In *Proceedings of the 9th International Conference on VLSI Design*, pages 178–85, 1996.

[2] M. A. Breuer. The effects of races, delays, and delay faults on test generation. *IEEE Transactions on Computers*, C-23(10):1078–1092, 1974.

[3] S. Chakraborty, D. Dill, and K. Yun. Min-max timing analysis and an application to asynchronous circuits. *Proceedings of the IEEE*, 87(2):332–346, 1999.

[4] T. J. Chakraborty, V. D. Agrawal, and M. L. Bushnell. Delay fault models and test generation for random logic sequential circuits. In *Proceedings of the 29th Design Automation Conference*, pages 165–172, 1992.

[5] J. E. Chen, C. L. Lee, and W. J. Shen. Single-fault fault-collapsing analysis in sequential logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(12):1559–1568, 1991.

[6] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, 1997.

[7] E. B. Eichelberger. Hazard detection in combinational and sequential switching circuits. *IBM Journal of Research and Development*, 9(2):90–99, 1965.

[8] H. K. Lee and D. S. Ha. Hope: An efficient parallel fault simulator for synchronous sequential circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(9):1048–1058, 1996.

[9] D. Muller and W. Bartky. A theory of asynchronous circuits. *In Annals of Computing Laboratory of Hardward University*, pages 204–243, 1959.

[10] C. J. Myers. *Asynchronous Circuit Design*. John Wiley and Sons, Inc., New York, 2001.

[11] O. Roig, J. Cortadella, M. A. Peiia, and E. Pastor. Automatic generation of synchronous test patterns for asynchronous circuits. In *Proceedings of the 34th Design Automation Conference*, pages 620–625, 1997.

[12] S. Sur-Kolay, M. Roncken, K. Stevens, P. P. Chaudhuri, and R. Roy. Fsimac: A fault simulator for asynchronous sequential circuits. In *Proceedings of the 9th Asian Test Symposium*, pages 114–119, 2000.