ESIM: A Multimodel Design Error and Fault Simulator for Logic Circuits

Hussain Al-Asaad
Computer Engineering Research Laboratory
Dept. of Electrical & Computer Engineering
University of California
One Shields Avenue, Davis, CA 95616-5294
E-mail: halasaad@ece.ucdavis.edu

John P. Hayes
Advanced Computer Architecture Laboratory
Dept. of Electrical Engineering & Computer Science
University of Michigan
1301 Beal Avenue, Ann Arbor, MI 48109-2122
E-mail: jhayes@eecs.umich.edu

Abstract

ESIM is a simulation tool that integrates logic fault and design error simulation for logic circuits. It targets several design error and fault models, and uses a novel mix of simulation algorithms based on parallel-pattern evaluation, multiple error activation, single fault propagation, and critical path tracing. Several experiments are discussed to demonstrate the power of ESIM.

1. Introduction

Fault simulation [3] consists of simulating a circuit's behavior in the presence of faults. Comparing the faulty response of the circuit to that of the fault-free response using the same test set *T*, we can determine the faults detected by *T*. Fault simulation has many applications, such as test set evaluation, fault-oriented test generation, fault dictionaries construction, and analysis of circuit operation in the presence of faults. There are many algorithms for fault simulation [3]: serial, parallel, deductive, concurrent, parallel-pattern single-fault propagation [14], and critical path tracing. Most of these algorithms target the single stuck-line (SSL) model for physical faults.

Recently, simulation-based design verification have received much attention [2][11][13]. An important part of the design verification process is design error simulation. Kang and Szygenda [13] present a simple parallel-pattern serial algorithm adopted from fault simulation. Developing efficient design error simulators is of great interest since the existence of such simulators may benefit many areas such as test generation for design errors [3] and design error diagnosis and correction [12][17].

Unlike SSL fault simulation where the number of faults is proportional to the number of nets in the circuit, design error simulation deals with a complex set of error models with the total number of errors proportional to $max\{2^p, p \times g^2\}$, where *g* is the number of gates and *p* is the maximum fanin of the gates in the circuit.

In this paper, we develop an efficient error/fault simulator ESIM that covers various new design error models as well as the older manufacturing fault models. ESIM is based on novel simulation algorithms that use a combination of parallel-pattern evaluation, multiple error activation, single fault propagation, and critical path tracing.

We discuss the design errors and logical faults that can be handled by ESIM in Section 2. We then review fault simulation methods for combinational circuits with emphasis on the parallel-pattern single fault propagation and critical path tracing in Section 3. We further discuss the application of critical path tracing to error simulation in this section. Section 4 discusses the implementation details of ESIM, presents the results of experiments performed using it, and presents preliminary results using a sequential version of ESIM. Finally, we draw some conclusions and suggest directions for further research.

2. Fault and design error models

Many types of faults and design errors have been classified in the literature [2][3][6][11][13]. These error types are not necessarily complete, but they are believed to be common in the lifetime of a digital system. In this paper, we consider several fault and error models which are described below.

Single Stuck-Line (SSL) Faults: The most widely-used logical fault model is the SSL model [3]. Under this model, every single signal line can become permanently fixed (stuck) at a logical 1 or 0 value. The model is simple and technology-independent. It represents some physical faults directly; more importantly, however, tests derived for SSL faults detect many actual design errors/faults. Since the number of SSL faults is proportional to the number of lines in the circuit, it is feasible to consider all possible SSL faults in medium scale designs.

Input Pattern (IP) Faults: Blanton and Hayes [8] presented a more general logical fault model called the *input pattern* (IP) fault model. Under this model, an IP fault in a module *M* changes the response of *M* to the input pattern *V* from F_V to $\bar{F}_V$. This IP fault is represented by $V'(F_V = \bar{F}_V)$. A *functional fault* in a module *M* changes the function implemented by *M*. It can be represented by a set of IP faults. The number of IP faults in a circuit *C* is proportional to $g \times 2^p$, where *g* is the number of gates in *C* and *p* is the maximum fanin of the gates in *C*.

Gate Substitution Errors (GSEs): According to experiments reported in [1], the most frequent error made in manual design is gate substitution, accounting for around 67% of all errors. Gate substitution refers to mistakenly replacing a gate *G* with another gate *G'* that has the same number of inputs. We represent this error by G/G'. For gates with multiple inputs, a *multiple-input GSE* (*MIGSE*) can have one of six possible forms: G/AND, G/NAND, G/OR, G/NOR, G/XOR, and G/XNOR. Each multiple-input gate can have five MIGSEs. For example, all MIGSEs can occur on an AND gate except G/AND which is not considered an error. For gates with a single input, i.e., buffers and inverters, a *single-input GSE* (*SIGSE*) can have one of two possible forms: G/NOT and G/BUF. Each single-input gate can have only one SIGSE. To cover extra/missing inverters in the gate substitution errors, a buffer can be inserted in each of a gate's fanout branches.

It has been suggested that most GSEs can be detected by a complete test set for SSL faults [2]. Our experimental results (Section 4) show that such a test set can cover 80% to 98% of MIGSEs and 100% of SIGSEs. The coverage of MIGSEs is a function of the circuit structure, as well as the types of gates used in the circuit.

Gate Count Errors (GCEs): We distinguish two types of gate count errors: extra-gate errors and missing-gate errors. An *extra-gate design error* (EGE) is defined as inserting a gate *G'* that has its *m* inputs taken from the *n* inputs of a gate *G* and feeding the output of *G'* to *G*. As a consequence, the number of inputs of gate *G* becomes $n - m + 1$. We represent an EGE by EG(*G'*,*G*). It is easy to see that EG(AND, AND), EG(AND, NAND), EG(OR, OR), EG(OR, NOR), EG(XOR, XOR), and EG(XOR, XNOR) are undetectable or redundant.

A *missing-gate design error* (MGE) is defined as removing a gate *G'* that has *m* inputs and feeds an *n*-input gate *G*, and then changing the inputs of *G'* into inputs of *G*; see Figure 1. As a consequence, the number of inputs of *G* becomes $N = n + m - 1$. We represent the MGE by MG(*G'*,*G*). As in the extra-gate case, the errors MG(AND, AND), MG(AND, NAND), MG(OR, OR), MG(OR, NOR), MG(XOR, XOR), and MG(XOR, XNOR) are undetectable.

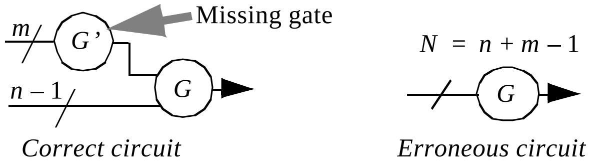

Figure 1 The missing-gate design error (MGE).

For $m = 2$, there are $\binom{N}{2}$ possible different configurations of the correct circuit shown in Figure 1. Since *G'* can change to five other gate types, we have a total of $5 \times \binom{N}{2}$ MGEs. For the general case of $m = i$, the total number of MGEs in an *N*-input gate is given by

$$5 \times \sum_{i=2}^{N-1} \binom{N}{i} = O(2^N)$$

Input Count Errors (ICEs) and Wrong Input Errors (WIEs): Input count errors are further classified into extra input and missing input errors. An *extra input design error* (EIE) is defined as the replacement of an *n*-input gate ($n \geq 2$) by an ($n + 1$)-input gate with the additional input connected to an arbitrary signal in the circuit. A *missing input design error* (MIE) is defined as the replacement of an *n*-input gate ($n \geq 3$) by an ($n - 1$)-input gate with its $n - 1$ inputs connected to an arbitrary subset of the original *n* inputs. We represent an EIE of a gate *G* by EI(*e*,*G*), where *e* is the extra input. We represent an MIE of a gate *G* by MI(*m*,*G*) where *m* is the source of the missing input. The number of ICEs in a circuit is very large—approximately $O(k^2)$, where *k* is the number of distinct signals in the circuit.

A *wrong input error* (WIE) is defined as a connection of a gate input to a wrong signal source. We represent a WIE on a gate *G* by WI(*u*,*w*,*G*), where *u* is the wrong input of the gate and *w* is the correct input. If a vector *v* detects WI(*u*,*w*,*G*), then it must set *u* and *w* to opposite values and propagate the signal at *u* to a primary output. The number of WIEs is larger than that of the ICEs, and it is approximately $O(k^2)$, where *k* is the number of distinct signals in the circuit. WIEs appears to be the second most common design error—around 17% of the errors reported in [1].

An important question concerning MIEs is the source of the missing input. It must not depend on the erroneous gate's output, otherwise, the circuit can become sequential. Errors that make a combinational circuit sequential can be detected by a levelization procedure [3]. Similarly, the source of wrong input of a WIE must not depend on the gate output.

Design Error Examples: Examples of the design errors discussed in this section are shown in Figure 2. The design errors of types SIGSE, MIGSE, EGE, MGE, and EIE are grouped in GP1. The remaining design errors, of type WIE and MIE, are combined in GP2. The number of errors/faults in GP1 for a typical circuit is proportional to the number of nets in the circuit. On the other hand, the number of errors in GP2 is proportional to the square of the number of lines in the circuit.

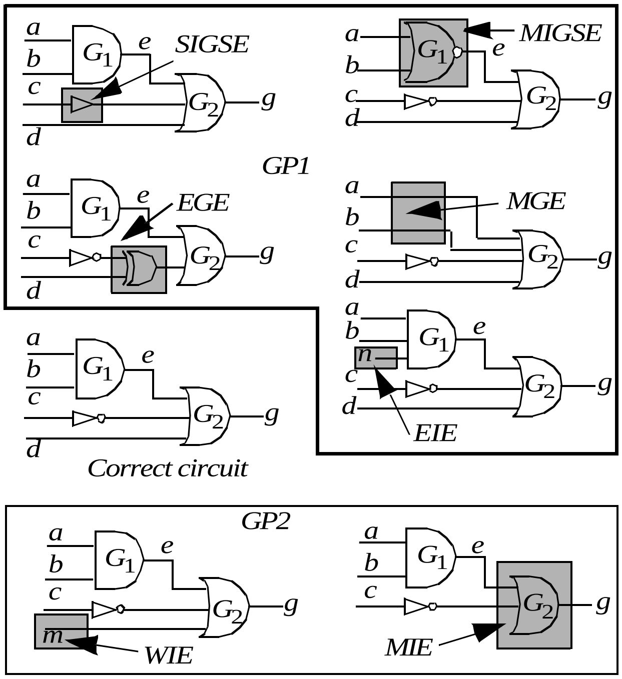

Figure 2 Examples of design errors.

3. The error and fault simulator ESIM

Many general approaches to fault simulation have been proposed such as serial, parallel [19], deductive [7], and concurrent[21]. Serial fault simulation is the slowest method of all, but uses the least amount of memory. It is based on simulating the fault-free circuit and the circuit in the presence of one fault, and then comparing the responses of the faulty and fault-free circuits. If the responses differ, then the fault is detected. The process is repeated for all the faults of interest, hence the execution time is proportional to the number of faults. Parallel fault simulation simulates a number W of faults simultaneously. Hence, it is faster than serial simulation but it needs more memory to deal with W faults at a time. Deductive and concurrent fault simulation techniques are based on detecting all possible faults in the circuit by a given test in one forward pass through the circuit. These methods are fast, but they have the disadvantage of unpredictable memory requirements [3]. The widespread use of design for testability techniques that transform a sequential circuit into a combinational one for testing purposes has increased the importance of specialized methods for combinational circuits.

Critical path tracing is a fault simulation method for combinational circuits that is based on simulating the fault-free circuit. It computes signal values for tracing paths from primary outputs towards primary inputs to determine the detected faults without explicitly computing the faulty signal values by gate evaluation or fault list processing. This method has received a lot of attention [3][4][14][16]. Its main advantage is that it directly identifies the faults detected by a test without simulating all possible faults. The details of this approach are described next.

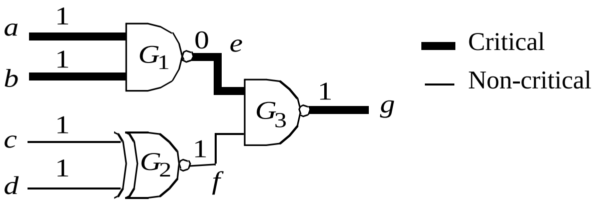

Figure 3 Critical path tracing in fanout-free circuits.

For every input vector, critical path tracing first simulates the fault-free circuit, then it identifies the detected faults by determining which signal values are *critical*. A line l has a critical value v in the test t iff t detects the fault l stuck-at-$\bar{v}$. After finding the critical lines in a test t, we can identify the SSL faults detected by t. The method starts by marking the primary outputs as critical. Then, it traverses the circuit backwards and determines the criticality of gate inputs according to the input values and the criticality of the gate output. For a gate $G \in \{AND, NAND, OR, NOR\}$, the following rules are employed:

- If only one input x has a controlling value and the output is critical, then x is critical.
- If all the inputs are non-controlling and the output is critical, then all inputs are critical.
- Otherwise, none of the inputs is critical.

For the other gates, $\{NOT, BUF, XOR, XNOR\}$, the inputs are critical if the output is critical. To illustrate, consider the fanout-free circuit in Figure 3. The signal values in response to 1111 are shown in the figure. The critical path tracing method starts by marking the output g as critical, then it examines the gate G_3. Since e is the only controlling input and the output is critical, then e is critical. The method then examines gate G_1 followed by G_2. For G_1, the inputs are non-controlling and the output is critical, hence the inputs a and b are critical. For G_2, the output is not critical, hence the inputs c and d are also not critical. After performing the critical path tracing analysis, the faults that are activated by the signal values and fall on a critical path are detected by the test 1111. Hence, the SSL faults a/0, b/0, e/1, and g/0 are detected by 1111.

For the case of circuits with fanout, we have to consider fanout stems, where a stem is a line that has multiple fanout. The criticality of the stem cannot be determined from its fanouts due to the fact that propagation of fault effects on multiple paths can block the propagation of the effects to primary outputs. The problem of determining the criticality of the stems is called *stem analysis*. The simplest solution to stem analysis is to explicitly simulate the stems to determine if they are critical. An efficient technique for stem analysis has not been found yet. Due to stem analysis, critical path tracing is not used alone in fault simulation. In [16], it determines the criticality of non-stem lines as described above, while the criticality of stems is determined by parallel-pattern single fault propagation (PPSFP). This approach combines two concepts: single-fault propagation and parallel-pattern evaluation.

- *Single-fault propagation* is a specialized serial fault simulation method for combinational circuits. Each SSL fault is injected in the circuit and the circuit is simulated; then the response is compared to the fault-free response. If they differ, the fault is detected. To speed this process, the faulty circuit is simulated starting at the fault site and continuing to the primary outputs. The gates at earlier levels than the fault site need not be evaluated because they are not changed.
- *Parallel-pattern evaluation* is a simulation technique that simulates *W* vectors concurrently, where *W* is in most cases the width of word in the host computer. Of course, this is only possible in combinational circuits where the order of applying the vectors is irrelevant. PPSFP starts by determining the fault-free response for *W* vectors. Then, a fault is injected and the faulty responses are computed. If the response of the circuit is different for any of the *W* vectors, then the fault is detected and the process is repeated for another fault. After checking for the last fault, another set of *W* vectors are selected and the process is repeated until either the vectors are exhausted or all the faults are detected.

Unlike the methods discussed above that target SSL faults only, ESIM is designed to efficiently fault simulate several different types of error and fault models including all those discussed in Section 2. The detection of an error/fault in a target circuit is determined by ESIM using the information about the criticality of the lines as well as the activation conditions for the faults/errors. A fault/error in a gate *G* is detected by a test *t* iff *t* activates the fault/error and the output of *G* is critical under *t*. Hence, if the output of a gate *G* is critical under a test *t*, then all the errors/faults at *G* that are activated by *t* are detected by it. The activation conditions for the faults/errors are summarized in Table 1. ESIM combines the following four techniques: parallel pattern evaluation where packets of 32 tests are simulated concurrently, multiple error and fault activation, single fault propagation at stems, and critical path tracing at the non-stem lines. We build explicit fault and error lists for SSLs, IPs, GSEs, GCEs, and EIEs. However, since the number of MIEs and WIEs is quadratic in the number of nets in the circuit, we use implicit partial error lists for these errors.

Table 1 Conditions for activating a fault/error by a test.

Fault/Error	Activation Condition
SSL *l*/*v*	*l*'s value is the complement of *v*
IP $V'(F_V = \bar{F}_V)$	The input pattern is *V*
SIGSE	None
MIGSE *G*/*G*'	Input pattern to *G* distinguishes between *G* and *G*'
EGE	Same as MIGSE
MGE MG(*G*',*G*)	Input pattern distinguishes between the existence and the absence of *G*'
EIE EI(*e*,*G*)	*e* is the only controlling input of *G*, if *G* is AND, NAND, OR, or NOR *e* is 1 if *G* is XOR or XNOR
MIE MI(*m*,*G*)	*m* is the only controlling input if *G* is AND, NAND, OR, or NOR *m* is 1 if *G* is XOR or XNOR
WIE WI(*u*,*w*,*G*)	*u* is critical and *w*'s value is the complement of that of *u*

```
/* C is the circuit*/
/* T is the simulation test set */
procedure GP1-Simulation( C,T);
begin
    Form the fault/error list L;
    Form stem list S;
    repeat
        Select a packet P of 32 tests from T;
        T := T - P;
        Set all signals in C to noncritical;
        S := S - {stems with no faults in their fanout-free regions;}
        Perform fault-free simulation using P;
        Determine criticality of stems in S;
        Traverse C backwards and determine criticality of
            all signals;
        Identify detected faults/errors D;
        L := L - D;
    until T is empty or L is empty;
    Output the results;
end;
```

Figure 4 Error simulation algorithm for GP1 errors.

ESIM is written using C++ in approximately 9000 lines of code. Its simulation algorithms for GP1 errors (GSEs, GCEs, and EIEs) and GP2 errors (MIES and WIEs) are shown in Figures 4 and 5, respectively. The simulation algorithms for SSL and IP faults are similar to that of GP1.

4. Experimental results

The major application of ESIM is to evaluate the coverage of design errors and logical faults by using various test sets that are determined by typical automatic test pattern generation tools such as the following:

- ATALANTA [15]: This is a combinational test pattern generator for SSL faults that is characterized by short test generation time as well as small test set size. It is based on the FAN algorithm [3] for test generation.
- RTESTS and ETESTS: These test pattern generators

```
procedure GP2-Simulation(C,T);
begin
    Form gate list GL;
    repeat
        Select a gate G from GL;
        GL := GL - {G};
        CT := T;
        repeat
            Select a packet P of 32 tests from CT;
            CT := CT - P;
            Perform fault-free simulation using P;
            Determine criticality of G's output;
            NC := {gates of C not in the cone of influence of G};
            if (G's output is critical) then
            repeat
                Select a gate G' from NC;
                NC := NC - {G'};
                Mark all detected MIEs and WIEs that have the
                    output of G' as the wrong source;
            until NC is empty;
        until CT is empty;
        Update the results;
    until GL is empty;
    Output the final results;
end;
```

Figure 5 Error simulation algorithm for GP2 errors.

were developed by us to produce random and exhaustive tests, respectively.

We now describe several experiments that illustrate the capabilities of ESIM. The circuits used in the experiments are the ISCAS-85 benchmark set [10] as well as few circuits from the 74X TTL IC series [20]. Table 2 shows the number of design errors and logical faults in these circuits.

- **Experiment 1** (*Exhaustive simulation*): The first experiment was conducted to investigate exhaustive simulation using tests generated by ETESTS. This experiment gives us the percentage of redundant design errors and logical faults in the simulated circuits. The results of the experiment are shown in Table 3, from which we see that the redundancy of some types of design errors can be as large as 11.6%, and that of IP faults can be as large as 33.5%. This experiment is performed only for those benchmarks where simulation with exhaustive tests is feasible—circuits with approximately 16 or fewer inputs.
- **Experiment 2** (*Random simulation*): The second experiment evaluates the random simulation approach. Random test sets of sizes 1 through 20 were generated by RTESTS for the c74283 carry-lookahead adder circuit and the coverage of design errors was determined using ESIM. The process was repeated 50 times and the average coverage obtained is shown in Table 4. The table shows that a small number of vectors provide good (but not full) coverage of design errors. The main problem with random simulation of this type is that it cannot guarantee high coverage with a relatively small number of vectors.

Table 2 Numbers of faults and design errors in the circuits used in the experiments.

Circuit	SSL faults	IP faults	GSEs		GCEs		ICEs		WIEs
			SIGSEs	MIGSEs	EGEs	MGEs	EIEs	MIEs	
c17	22	24	11	30	2	0	12	40	92
c432	524	2508	312	600	67	9460	296	18482	52063
c499	758	1072	337	810	104	1500	368	31452	81576
c880	942	1614	586	1470	199	1040	640	120779	299868
c1355	1574	2384	881	2370	216	1500	992	208476	480408
c1908	1879	5374	1467	2205	252	12775	1059	358816	1217410
c2670	2747	4842	1994	3380	476	4485	1559	940307	2881417
c3540	3428	10258	2584	4780	634	23470	2226	1513437	4658069
c5315	5350	11728	3902	7065	986	18110	3492	3454806	10738696
c6288	7744	9600	3904	11920	944	0	4768	4999155	10055805
c7552	7550	14636	5450	10510	1408	14390	4734	7707830	22536439
7485	137	472	86	155	20	1565	97	974	3456
74181	237	454	146	265	36	855	143	3750	11621
74283	128	240	74	150	17	460	76	985	3285

Table 3 The coverage of faults and design errors using exhaustive test sets.

Circuit	Test set size	Detected SSL faults	Detected IP faults	Detected GSEs		Detected GCEs		Detected ICEs		Detected WIEs
				SIGSEs	MIGSEs	EGEs	MGEs	EIEs	MIEs	
c17	32	100	100	100	100	100	n/a	100	95.0	100
7485	2048	100	66.5	100	88.4	100	94.4	100	91.2	97.5
74181	16384	100	96.5	100	98.5	88.9	99.5	100	96.6	99.1
74283	512	100	96.7	100	94.7	100	100	100	90.0	96.9

Table 4 The coverage of SSL faults and design errors in the 74283 adder using random test sets.

Test set size	SSL	SIGSE	MIGSE	EGE	MGE	EIE	MIE	WIE
1	28.5	47.7	48.1	60.9	23.1	17.3	17.7	23.0
2	44.9	61.1	65.2	76.7	35.6	30.2	29.5	38.0
3	54.1	68.3	72.5	86.4	45.7	39.3	36.3	47.2
4	63.8	75.6	80.6	93.4	54.1	48.5	45.3	57.1
5	67.7	78.1	81.9	92.7	56.2	53.1	49.4	60.5
6	72.1	81.0	86.5	96.7	61.3	57.7	54.5	65.6
7	73.6	81.8	87.5	96.5	62.5	59.9	57.1	67.8
8	78.1	86.9	89.3	99.1	70.2	66.9	60.2	72.8
9	76.9	84.5	88.9	97.9	69.7	66.6	60.2	72.2
10	80.9	87.6	90.7	99.8	72.1	70.4	65.8	75.8
11	81.1	87.4	91.4	99.8	74.0	73.2	65.5	76.5
12	83.2	89.6	91.3	99.8	76.0	74.9	68.2	78.6
13	84.0	90.7	92.5	100.0	78.7	75.2	70.4	79.5
14	82.6	90.2	92.1	100.0	79.8	76.6	68.4	78.9
15	85.1	92.0	92.5	100.0	80.7	78.1	71.3	81.0
16	85.5	92.4	93.0	100.0	82.2	78.1	71.8	81.5
17	85.2	91.6	92.7	100.0	80.9	79.3	71.7	81.3
18	87.9	94.4	93.2	100.0	84.7	81.2	74.8	83.9
19	87.4	93.1	93.0	100.0	83.9	81.7	74.8	83.3
20	87.9	93.4	93.3	100.0	84.6	82.2	74.8	83.9

Table 6 ESIM design error simulation time using ATALANTA's complete SSL tests.

Circuit	Initialization		Simulation for GP1		Simulation for GP2		Total time[a]
	Time	%	Time	%	Time	%	
c17	0.04	50.0	0.02	25.0	0.02	25.0	0.08
c432	0.26	1.1	15.81	67.6	7.31	31.3	23.38
c499	0.34	2.3	3.70	24.9	10.81	72.8	14.85
c880	0.55	1.5	1.88	5.3	33.14	93.2	35.57
c1355	0.86	0.7	7.33	6.1	112.84	93.2	121.03
c1908	1.36	0.4	25.41	9.2	250.91	90.4	277.68
c2670	1.99	0.4	18.13	3.5	493.21	96.1	513.33
c3540	2.96	0.2	218.99	16.1	1140.77	83.7	1362.71
c5315	6.38	0.3	89.50	3.9	2199.19	95.8	2295.07
c6288	5.25	0.3	73.76	3.9	1824.62	95.8	1903.63
c7552	8.35	0.1	179.70	2.5	7079.72	97.4	7267.77
7485	0.08	3.6	1.85	82.2	0.32	14.2	2.25
74181	0.12	7.7	0.51	32.7	0.93	59.6	1.56
74283	0.07	11.7	0.29	48.3	0.24	40.0	0.60

[a] In seconds on a SUN SPARC 20.

- **Experiment 3** (*Simulation using SSL tests*): A third experiment was conducted to determine the coverage of design errors and logical faults using tests for SSL faults. The effectiveness of a complete test set for SSL faults (determined by ATALANTA) in detecting design errors is shown in Tables 4 and 5. As discussed earlier, most of the simulation time is spent in the simulation of GP2, especially as the circuits become larger. The effectiveness of the complete test sets for SSL faults in detecting IP faults is shown in Table 7. The results show that complete test sets for SSL faults do a very poor job in detecting IP faults.

Although ESIM was designed to handle the simulation of design errors and logical faults, it has capabilities that can be used in other applications.

- *Test grading*: The error simulator can determine the number of faults detected by a given test. This information is useful in applications such as hardware test generation and test set compaction.
- *Fault and error grading*: This refers to classifying the faults and errors as hard-to-detect (also called random pattern resistant) or easy-to-detect. Fault and error grading has many applications such as test generation and test point selection.
- *Fault table generation*: ESIM can generate a complete fault table for circuits with 16 or fewer inputs; for larger circuits, partial fault tables can also be generated.

Table 5 The coverage of SSL faults and design errors using ATALANTA's complete SSL tests.

Circuit	Test set size	Detected SSL faults	Detected GSEs		Detected GCEs		Detected ICEs		Detected WIEs
			SIGSEs	MIGSEs	EGEs	MGEs	EIEs	MIEs	
c17	5	100.0	100.0	80.0	100.0	n/a	100.0	57.5	88.0
c432	46	99.2	100.0	89.3	100.0	95.5	98.7	71.3	96.4
c499	52	98.9	100.0	97.8	46.2	89.6	97.8	88.8	98.6
c880	47	100.0	100.0	90.3	100.0	94.6	100.0	84.9	98.6
c1355	85	99.5	100.0	82.0	100.0	89.6	99.2	82.2	98.6
c1908	115	99.5	100.0	84.7	97.6	88.7	99.2	85.8	97.0
c2670	106	95.7	99.7	86.5	87.6	88.9	93.2	85.9	97.4
c3540	152	96.0	99.3	89.5	90.5	81.2	94.2	82.7	97.5
c5315	106	98.9	100.0	89.5	98.9	91.7	98.3	94.5	98.9
c6288	35	99.6	99.6	85.6	100.0	n/a	99.3	89.3	99.6
c7552	199	98.3	100.0	86.6	97.4	90.3	97.2	93.2	98.7
7485	25	100.0	100	88.4	100.0	89.8	100.0	83.4	92.7
74181	18	100.0	100	96.2	88.9	90.6	100.0	81.7	94.0
74283	12	100.0	100	91.3	100	84.1	100.0	74.5	92.2

Table 7 The coverage of IP faults using ATALANTA's complete SSL tests.

Circuit	Test set size	Detected IP faults	Simulation time[a]
c17	5	75.0	0.0
c432	46	25.7	1.0
c499	52	80.8	1.1
c880	47	85.3	3.8
c1355	85	75.3	6.2
c1908	115	61.9	17.3
c2670	106	76.5	21.4
c3540	152	50.3	79.1
c5315	106	74.3	183.1
c6288	35	81.7	219.1
c7552	199	78.5	391.8
7485	25	40.7	0.1
74181	18	70.7	0.1
74283	12	56.7	0.1

[a] In seconds on a SUN SPARC 20.

This feature of the simulator is used to analyze the faults of a given circuit. Note that ESIM performs simple SSL fault collapsing to reduce the size of the fault table.

- *Test generation*: The simulator also supports a fast, greedy test generation algorithm based on covering the fault table. Experimental results show that the algorithm often produces near-minimal test sets in circuits with a small number of inputs.
- *Dependency evaluation*: This refers to the structural dependency between any two circuit outputs. The idea is to determine the common lines in the cones of influence of two outputs. This is useful in on-line testing, where circuit outputs are compacted to decrease the hardware overhead of the hardware test/signature generator.
- *Netlist translation*: Most CAD tools accept various netlist formats, such as ISCAS-85 and BLIF. ESIM can translate any ISCAS-85 description to ISCAS-89 [9], BLIF, and Verilog. A major use of the translator is in the synthesis of logic circuits using SIS [18], whose input format is BLIF.

ESIM also reports some statistics about the circuit being simulated. This can be seen by the sample run shown in Figure 6, where ESIM determines the coverage of an exhaustive test set for 74283, a 4-bit carry-lookahead adder circuit.

It is difficult to compare the results obtained by ESIM to related work in the literature for several reasons: (1) different error models are used; (2) test set sizes are missing from the results of [13]; and (3) standard benchmarks are not used in most prior work.

ESIM is useful not only for combinational but also for sequential circuits. Preliminary experimental results using a sequential version of ESIM on a subset of non-scan sequential benchmarks from the ISCAS-89 suite [9] are shown in Table 8. The test sequence *S* used in the simulation was generated in [5] to detect the design error models in GP1. In addition to computing the coverage of design error models in GP1 and GP2, ESIM returns the coverage of extra latch errors (ELEs) and missing latch errors (MLEs). The coverage of design errors is high for all circuits, except for s420 whose internal nets have low controllability and observability.

```
esim c74283.isc c74283.xhv

Gates = 104
=================================
  Gtype Ngates  Mxfin    Mxfout
=================================
  nand    4       2        7
  and    14       5        1
  nor     8       5        5
  or      0       0        0
  xor     4       2        0
  xnor    0       0        0
  inpt    9       0        2
  from   59       1        1
  not     6       1        5
  buff    0       0        0
=================================
Levels  = 6
Inputs  = 9
Outputs = 5
Stems   = 22
Tests   = 512

  SSL     128     128    100.00   0.09
======= ======= ======= ======= ======
 SIGSE     74      74    100.00
 MIGSE    150     142     94.67
  EGE      17      17    100.00
  MGE     460     460    100.00
  EIE      76      76    100.00
 TOTAL    777     769     98.97   0.22
======= ======= ======= ======= ======
  MIE    1143    1029     90.03
  WIE    3285    3184     96.93
 TOTAL   4428    4213     95.14   2.86
======= ======= ======= ======= ======
  IP      240     232     96.67   0.15
======= ======= ======= ======= ======

Initialization Time = 0.27
Simulation Time     = 3.32
Total Time          = 3.59
```

Figure 6 Output generated by a sample run of ESIM.

5. Discussion

ESIM is based on a novel combination of parallel-pattern evaluation, multiple fault/error activation, single fault propagation, and critical path tracing. It can handle several types of design errors and logical faults, and can readily be extended to cover additional error/fault models. The experiments reported here show that ESIM is relatively fast. They also confirm a number of interesting observations made before [2][6] such as: (*i*) most design errors and logical faults can be covered by small test sets, (*ii*) the percentage of redundant design errors and IP faults is large in some circuits, and (*iii*) complete test sets for SSL faults are reasonably good tests for simulation-based design verifica-

Table 8 The coverage of SSL faults and design errors in sequential benchmarks.

Circuit	Size of *S*	Detected errors/faults									
		SSL	SIGSE	MIGSE	EGE	MGE	EIE	MIE	WIE	ELE	MLE
s27	49	100.0	100.0	95.0	100.0	n/a	100.0	74.7	94.4	100.0	100.0
s208	448	95.6	91.3	87.9	87.8	83.8	91.6	72.4	93.0	87.5	81.8
s298	448	86.4	91.1	93.6	100.0	96.92	77.5	67.5	84.6	100.0	100.0
s344	264	93.9	97.3	90.7	100.0	85.0	91.9	76.9	94.2	100.0	95.0
s349	352	94.9	97.7	90.8	100.0	85.0	93.5	79.9	95.9	100.0	95.0
s386	500	85.1	97.1	92.7	78.4	98.2	69.6	67.1	87.6	100.0	92.9
s420	499	54.5	58.1	69.4	71.9	48.1	51.8	32.8	52.8	81.3	36.8
s641	360	87.6	93.5	94.8	73.5	90.6	81.7	65.2	90.4	78.9	89.8

tion.

Several aspects of ESIM's error modeling and simulation capabilities remain to be investigated, especially in the case of sequential circuits. Its overall performance could be improved by introducing error collapsing. Finding ways to collapse the number of the missing or wrong inputs that need to be considered would be especially useful. As Table 6 shows, most of ESIM's simulation time is spent on MIEs and WIEs. The relation between design errors and IP faults also seems worth exploring since tests for IP faults appear to cover many design error types including hard-to-model errors, as well as unknown manufacturing fault types [8].

Acknowledgment

This research was supported by the National Science Foundation under Grant No. CCR-9872066.

References

[1] E. J. Aas, T. Steen, and K. Klingsheim, "Quantifying design quality through design experiments", *IEEE Design and Test*, Vol. 11, No. 1, pp. 27-37, Spring 1994.

[2] M. S. Abadir, J. Ferguson, and T. E. Kirkland, "Logic design verification via test generation", *IEEE Transactions on Computer-Aided Design*, Vol. 7, pp. 138-148, January 1988.

[3] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, New York, 1990.

[4] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical path tracing: An alternative to fault simulation", *IEEE Design and Test of Computers*, Vol. 1, No. 1, pp. 83-93, February 1984.

[5] H. Al-Asaad, *Lifetime Validation of Digital Systems via Fault Modeling and Test Generation*, Ph.D. Dissertation, University of Michigan, 1998.

[6] H. Al-Asaad and J. P. Hayes, "Design verification via simulation and automatic test pattern generation", *Proc. International Conference on Computer-Aided Design*, 1995, pp. 174-180.

[7] D. B. Armstrong, "A deductive method of simulating faults in logic circuits", *IEEE Transactions on Computers*, Vol. C-21, pp. 464-471, May 1972.

[8] R. D. Blanton and J. P. Hayes, "Properties of the input pattern fault model", *Proc. International Conference on Computer Design*, 1997, pp. 372-380.

[9] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits", *Proc. International Symposium on Circuits and Systems*, 1989, pp. 1929-1934.

[10] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran", *Proc. International Symposium on Circuits and Systems*, 1985, pp. 695-698.

[11] B. Chen, C. L. Lee, and J. E. Chen, "Design verification by using universal test sets", *Proc. Third Asian Test Symposium*, 1994, pp. 261-266.

[12] S.-Y. Huang et al., "ErrorTracer: A fault simulation-based approach to design error diagnosis", *Proc. International Test Conference*, 1997, pp. 974-981.

[13] S. Kang and S. A. Szygenda, "The simulation automation system (SAS); Concepts, implementation, and results", *IEEE Transactions on VLSI Systems*, Vol 2, No. 1, pp. 89-99, March 1994.

[14] H. K. Lee and D. S. Ha, "An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation", *Proc. International Test Conference*, 1991, pp. 946-955.

[15] H. K. Lee and D. S. Ha, "On the generation of test patterns for combinational circuits", Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Tech. Rep. 12-93, 1993.

[16] F. Maamari and J. Rajski, "A method of fault simulation based on stem regions", *IEEE Transactions on Computer-Aided Design*, Vol. 9, pp. 212-220, February 1990.

[17] D. Nayak and D. M. H. Walker, "Simulation-based design error diagnosis and correction in combinational digital circuits", *Proc. VLSI Test Symposium*, 1999, pp. 70-78.

[18] E. M. Sentovich et al., "SIS: A system for sequential circuit synthesis", Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, Memorandum No. UCB/ERL M92/41, May 1992.

[19] S. Seshu, "On an improved diagnosis program", *IEEE Transactions on Electronic Computers*, Vol. EC-12, pp. 76-79, February 1965.

[20] Texas Instruments, *The TTL Logic Data Book*, Dallas, 1988.

[21] E. G. Ulrich and T. G. Baker, "Concurrent simulation of nearly identical digital networks", *IEEE Computer*, Vol. 7, pp. 39-44, April 1974.

[22] D. Van Campenhout, H. Al-Asaad, J. P. Hayes, T. Mudge, and R. Brown, "High-level design verification of microprocessors via error modeling", *ACM Transactions on Design Automation of Electronic Systems*, Vol. 3, No. 4, pp. 581-599, October 1998.