PROOFS: A SUPER FAST FAULT SIMULATOR FOR SEQUENTIAL CIRCUITS

Wu-Tung Cheng
AT&T Bell Laboratories
P.O. Box 900
Princeton, NJ 08540

Janak H. Patel
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

ABSTRACT

This paper describes PROOFS, a super fast fault simulator for synchronous sequential logic circuits. PROOFS achieves high performance by combining all the advantages in differential fault simulation, single fault propagation, and parallel fault simulation to minimize the memory requirements, to reduce events that need to be simulated, and to simplify the complexity of the software implementation. The experimental results of PROOFS and other available fault simulators on 20 benchmark circuits showed that PROOFS is the best.

I. INTRODUCTION

With the development of VLSI technologies, test sequences with very high fault coverage have become increasingly important in order to maintain acceptable field reject rates [1]. Fault simulators are used to determine which faults are detected by a test sequence. The fault coverage information not only represents the quality of the test sequence but also speeds up the test generation process. After a test sequence is generated for a target fault using a time consuming test generator, a fault simulator is usually used for finding other faults that are also detected. In this manner, the number of faults for which tests are attempted by the test generator can be dramatically reduced.

The single stuck-at fault model has been very successfully used in many contemporary fault simulators. Therefore only single stuck-at faults are considered in this paper. In fault simulation, each test pattern is run with the good machine and every faulty machine which has a line staying fixed at a high voltage (a stuck-at-1 fault) or staying fixed at a low voltage (a stuck-at-0 fault). If the output patterns of a faulty machine and the good machine are different, the corresponding fault is said to have been detected. As shown in Table 1, there are \( m \) faulty machines and the test sequence has \( n \) test vectors. Each column corresponds to a test vector and each row corresponds to the good machine or one of the bad machines. An entry in the table gives the machine status which consists of the logic values of all lines in the corresponding machine after the specific test vector has been applied. The task of fault simulation is to generate the primary output values for each one of the \((m+1)n\) machine status. Since these machine status are very similar, two strategies based on event-driven simulation have been used to speed up the fault simulation process.

The conventional strategy of fault simulation is to generate each machine status from its left neighbor in the same row (same machine, different test vector). For example, in Table 1, we can generate \( B_{1,i+1} \) from \( G_{i} \) by simulating the differences of their test vectors \( V_{i+1} \) and \( V_{i} \) as the initial event sources and the effects (events) inside the circuit. Fault simulation approaches using this strategy are serial fault simulation, parallel fault simulation [2], deductive fault simulation [3], and concurrent fault simulation [4]. Among these approaches, concurrent fault simulation offers the fastest speed [5].

The newer strategy is to generate each machine status from its reference machine status in the same column (same test vector, different machine). This strategy was first used exclusively for combinational circuits in testdetect [6] (which was later called single fault propagation [7]). Since single fault propagation always uses the good machine as the reference machine, it simulates only the fault effects of each faulty machine from the good machine. For example, in Table 1, \( B_{1,i+1} \) is generated from \( G_{i+1} \) by simulating the fault site of \( B_{1} \) as the initial event source. In general, \( B_{1,i+1} \) resembles \( G_{i+1} \) more than \( B_{1,i} \), therefore this strategy has fewer events than serial fault simulation. However, to restore the status of the good machine before every faulty machine simulation is the performance overhead. Single fault propagation was further improved by Waicukauski [8] to use the full length of a computer word for simulation of a machine with several input vectors in parallel. This is known as the PPSFP (Parallel Pattern Single Fault Propagation) technique. Furthermore, there are efficient heuristics proposed in [9-12] to trace fault effects in combinational circuits, and thus only small number of faulty machines need to be explicitly simulated. In general, for combinational circuits, single fault propagation is faster.

<table>
<thead>
<tr>
<th>( V_1 )</th>
<th>( V_i )</th>
<th>( V_{i+1} )</th>
<th>( V_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>( G_1 )</td>
<td>( G_i )</td>
<td>( G_{i+1} )</td>
</tr>
<tr>
<td>Bad_1</td>
<td>( B_{1,1} )</td>
<td>( B_{1,i} )</td>
<td>( B_{1,i+1} )</td>
</tr>
<tr>
<td>Bad_2</td>
<td>( B_{2,1} )</td>
<td>( B_{2,i} )</td>
<td>( B_{2,i+1} )</td>
</tr>
<tr>
<td>Bad_d</td>
<td>( B_{d,1} )</td>
<td>( B_{d,i} )</td>
<td>( B_{d,i+1} )</td>
</tr>
<tr>
<td>Bad_s</td>
<td>( B_{s,1} )</td>
<td>( B_{s,i} )</td>
<td>( B_{s,i+1} )</td>
</tr>
<tr>
<td>Bad_m</td>
<td>( B_{m,1} )</td>
<td>( B_{m,i} )</td>
<td>( B_{m,i+1} )</td>
</tr>
</tbody>
</table>
than concurrent fault simulation [13].

Recently, differential fault simulation, DSIM, was proposed in [14] to extend the application of the new strategy to synchronous sequential circuits. For sequential circuits, to generate $B_{k+1}$, from $G_i$, not only the fault site but also the value differences at the state elements have to be simulated as the initial event sources. Based on event-driven simulation, DSIM can record the value differences at the state elements with little overhead [14]. Because a fault in sequential circuits can be generated in several time frames and can be propagated across time frames, it is difficult to trace them and none of the efficient heuristics proposed in [9-12] are useful. Therefore, every faulty machine has to be simulated explicitly and the overhead to restore the status of the good machine before every faulty machine simulation becomes essential. In order to eliminate this overhead, the reference machine in DSIM is not the good machine but the previous machine just simulated. For example, $B_{k+1,i}$ in Table 1 is not generated from $G_i$ but from $B_{k,i}$. DSIM was shown in [14] to be 3-12 times faster than an existing concurrent fault simulator [15] on 20 benchmark circuits [16,17].

In [14], another fault simulator, SSIM, was also implemented for sequential circuits. The difference between DSIM and SSIM is that if the reference machine in DSIM is the machine just simulated while the reference machine in SSIM is always the good machine. It is clear that the ordering of the fault list affects the performance of DSIM but not the performance of SSIM. Four different ordering strategies tried for DSIM in [14] showed that they all had more events than SSIM had. Unfortunately, the method to restore the good machine status before every bad machine simulation in SSIM has an unacceptable time penalty. This result suggested that the fault ordering in DSIM should be improved or the overhead to restore the good machine status in SSIM should be improved.

In this paper, a technique which can restore the good machine status before every faulty machine simulation with little time penalty is proposed. This technique will be described in Section II. Based on this technique, we developed a new fault simulator, ROOFS, which has fewer events and is faster than DSIM. ROOFS was further improved by using the full length of a computer word for a parallel simulation of several machine status in the same column of Table 1 (same test vector, different machines). This fault simulator is called PROOFS and will be described in Section III. The conclusions are discussed in Section IV.

There are other approximate fault simulation approaches that trade the absolute accuracy of the results for a significant reduction in computation time. Examples of these methods are fast fault grading [18], critical path tracing [19], and statistical analysis [20]. In this paper, however, we only deal with exact fault simulation.

II. RESTORATION TECHNIQUE

Our method to restore the good machine status before every faulty machine simulation requires three different copies of the circuit status simultaneously. The first copy always records the good machine status (the status in the first row of Table 1), which are generated as true-value simulation. Based on the status in the first copy, every faulty machine status is generated and recorded in the second copy and the third copy as the following procedure. During the generation of every faulty machine status, an unique simulation ID is given. If any value is changed from the good machine value, the new value is recorded on the second copy and the simulation ID is marked on the third copy. That means each line has three values: the good machine value in the first copy, the faulty machine value in the second copy, and an associated ID in the third copy. If one line has an associated ID in the third copy which is the same as the simulation ID, the value of this line is the value in the second copy, otherwise its value is the value in the first copy. Since the good machine status is always available in the first copy, there is no time penalty to restore it before every faulty machine simulation. The requirement of this technique is that all simulation ID used for the faulty machine simulation must be unique. Thus, the third copy should be clean up before any ID has to be reused. Since our ID is recorded in a 32 bit word, more than 4 billion faulty machine simulation runs can be processed without clean up. Therefore the time penalty is very little.

Based on this restoration technique, a new fault simulation algorithm is proposed in Figure 1. It is much simpler than DSIM in [14]. The new implementation is called RestOrative Order-independent Fault Simulator, ROOFS.

Because of the simplicity of the new algorithm, ROOFS is very easy to implement and maintain. As DSIM, ROOFS was also implemented as an integrated part of an existing sequential circuit test generator program, STG3 [21,22]. The experiments in [14] were repeated here to compare the performance of ROOFS and DSIM. Table 2 shows the characteristics of 10 combinational benchmark circuits [16] and 10 sequential benchmark circuits [17]. The fault lists include only the faults remaining after fault collapsing. The test vectors were generated by using STG3 [23]. The experimental results of DSIM and ROOFS are shown in Table 3. The time is in seconds. ROOFS has 15% to 46% fewer events and 21% to 54% time reduction. The reasons for more time reduction than event reduction is the simpler implementation in ROOFS.

For every test vector

```c
    do true-value simulation;

    for every undetected faulty machine
      do event-driven simulation;
    if the fault is detected, drop the fault;
  }
```

Figure 1. The Algorithm of ROOFS.
III. PARALLEL MACHINE SIMULATION

It is well known that logic gate operation can be simulated by using logic bit operations in computer instructions. Since every bit in one word can be executed simultaneously, 32 (word length) logic gate operations can be simulated simultaneously also. This technique is called parallel simulation. The performance of ROOFS can be improved by using parallel simulation. Unlike parallel pattern single fault propagation [8] which simulates one machine with 32 test vectors in parallel for combinational circuits, we simulate 32 machines with the same test vector in parallel for sequential circuits. The fault simulator implemented is called PROOFS (Parallel ROOFS). It is obvious that how to group faulty machines affects the performance of PROOFS. The ideal fault grouping should put 32 faulty machines with similar events in one group. This is similar to the fault ordering problem in DSIM. PROOFS uses the same fault ordering as DSIM which is depth first from primary outputs to primary inputs and puts every 32 consecutive faulty machines in one group.

The coding method in [24] for three values is extended here. Since our application is for sequential circuits, four values are used. They are 0, 1, X (unknown), and Z (high impedance). Z is used for circuits with BUS nodes to identify bus conflicts. Here, if more than one inputs has non high impedance value, it is called a bus conflict. To code these four values, two bits have to be used. 0 is coded as (1,0), 1 is coded as (0,1), X is coded as (0,0) and Z is coded as (1,1). The first bit is called 0-bit and the second bit is called 1-bit. Two words are used to represent the group value of each line. Each group has 32 machines. The first word is called 0-word which contains all the 0-bits of these 32 machines. The second word is called 1-word which contains all the 1-bits of these 32 machines. With this coding method, 32 machines can be simulated simultaneously for all the logic functions, as shown in Table 4. In Table 4, & is logic AND instruction and I is logic OR instruction. A and B are the inputs and C is the output. For AND gates, OR gates and BUS nodes, if they have more than two inputs, their respective instructions in Table 4 have to be repeated for every extra input. For tri-state gate TRIG, A is the data input and B is the control input. Here, we have the restriction that the output of every tri-state gate must be connected to a BUS node, and the inputs of a bus node must be from primary inputs or tri-state gates. Also, if the output value of a BUS node is Z, it is treated as X. With these arrangements, we do not have to deal with Z in functions AND, OR, NOT, and EXOR. For BUS nodes, the operation \((A_0&B_0)(A_0&B_1)(A_1&B_0)(A_1&B_0)\) is carried out for the good machine to identify bus conflicts. For faulty machines, this operation is skipped for their outputs will be given value X when bus conflicts happen.

The previous experiments were repeated again to show the speed-up of PROOFS. The experimental results are shown in Table 5. Compared with ROOFS, PROOFS has 12% to 67% fewer events and 13% to 65% time reduction. The search for a better fault grouping for PROOFS is under investigation.

IV. CONCLUSIONS

A little penalty technique to restore the good machine status before every faulty machine simulation was developed in this paper. Based on this technique, a new fault simulator ROOFS was modified from a differential fault simulator DSIM. Compared with DSIM, ROOFS reduces events by 15%
Table 3. Experimental Results of DSIM and ROOFS.

<table>
<thead>
<tr>
<th></th>
<th>DSIM</th>
<th>ROOFS</th>
<th>ratio</th>
<th>DSIM</th>
<th>ROOFS</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>c432</td>
<td>4853</td>
<td>3470</td>
<td>0.72</td>
<td>10.48</td>
<td>7.13</td>
<td>0.88</td>
</tr>
<tr>
<td>c499</td>
<td>3977</td>
<td>3314</td>
<td>0.84</td>
<td>7.70</td>
<td>5.82</td>
<td>0.76</td>
</tr>
<tr>
<td>c880</td>
<td>6971</td>
<td>5073</td>
<td>0.73</td>
<td>15.63</td>
<td>11.07</td>
<td>0.71</td>
</tr>
<tr>
<td>c1355</td>
<td>12168</td>
<td>10232</td>
<td>0.84</td>
<td>30.60</td>
<td>23.63</td>
<td>0.77</td>
</tr>
<tr>
<td>c1908</td>
<td>40835</td>
<td>34428</td>
<td>0.84</td>
<td>81.80</td>
<td>61.40</td>
<td>0.75</td>
</tr>
<tr>
<td>c2670</td>
<td>39077</td>
<td>29259</td>
<td>0.75</td>
<td>95.97</td>
<td>67.95</td>
<td>0.71</td>
</tr>
<tr>
<td>c3540</td>
<td>73801</td>
<td>53473</td>
<td>0.72</td>
<td>152.55</td>
<td>99.15</td>
<td>0.65</td>
</tr>
<tr>
<td>c5315</td>
<td>1153006</td>
<td>871832</td>
<td>0.76</td>
<td>260.43</td>
<td>189.12</td>
<td>0.73</td>
</tr>
<tr>
<td>c6288</td>
<td>1348622</td>
<td>817484</td>
<td>0.78</td>
<td>176.95</td>
<td>116.14</td>
<td>0.66</td>
</tr>
<tr>
<td>c7552</td>
<td>1653505</td>
<td>1362453</td>
<td>0.82</td>
<td>356.63</td>
<td>282.02</td>
<td>0.79</td>
</tr>
<tr>
<td>s298</td>
<td>22133</td>
<td>185745</td>
<td>0.76</td>
<td>42.70</td>
<td>25.77</td>
<td>0.60</td>
</tr>
<tr>
<td>s349</td>
<td>122521</td>
<td>101116</td>
<td>0.83</td>
<td>22.93</td>
<td>15.98</td>
<td>0.70</td>
</tr>
<tr>
<td>s526</td>
<td>3584123</td>
<td>2902862</td>
<td>0.81</td>
<td>684.87</td>
<td>456.12</td>
<td>0.67</td>
</tr>
<tr>
<td>s713</td>
<td>118897</td>
<td>83017</td>
<td>0.70</td>
<td>23.67</td>
<td>16.22</td>
<td>0.69</td>
</tr>
<tr>
<td>s832</td>
<td>909915</td>
<td>590202</td>
<td>0.56</td>
<td>181.32</td>
<td>96.72</td>
<td>0.53</td>
</tr>
<tr>
<td>s338</td>
<td>785897</td>
<td>665868</td>
<td>0.85</td>
<td>139.38</td>
<td>108.82</td>
<td>0.78</td>
</tr>
<tr>
<td>s1238</td>
<td>337894</td>
<td>243096</td>
<td>0.72</td>
<td>78.50</td>
<td>60.95</td>
<td>0.78</td>
</tr>
<tr>
<td>s1494</td>
<td>3292979</td>
<td>1770375</td>
<td>0.54</td>
<td>581.80</td>
<td>283.40</td>
<td>0.49</td>
</tr>
<tr>
<td>s5378</td>
<td>4789093</td>
<td>3188203</td>
<td>0.67</td>
<td>967.38</td>
<td>619.07</td>
<td>0.64</td>
</tr>
<tr>
<td>s35932</td>
<td>14599275</td>
<td>11428644</td>
<td>0.78</td>
<td>2863.45</td>
<td>1999.20</td>
<td>0.70</td>
</tr>
</tbody>
</table>

to 46% and time by 21% to 51%. Combining ROOFS with parallel simulation technique, PROOFS further reduces events by 12% to 67% and time by 13% to 65%. To search for a better fault ordering for DSIM and a better fault grouping for PROOFS is the future research.

ACKNOWLEDGMENT

This research was performed at the University of Illinois, Urbana-Champaign, while Wu-Tung Cheng was a Visiting Research Assistant Professor. The work is supported in part by AT&T, and in part by Semiconductor Research Corporation under contract SRC-88-DP-109.

REFERENCES


Table 4. Function Evaluations.

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>A1</td>
<td>A1</td>
<td>A1</td>
<td>A2</td>
<td>A2</td>
<td>A3</td>
<td>A4</td>
<td>A5</td>
<td>A6</td>
<td>A7</td>
<td>A8</td>
<td>A9</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
<td>B4</td>
<td>B5</td>
</tr>
<tr>
<td>OR</td>
<td>A1</td>
<td>A2</td>
<td>A3</td>
<td>A4</td>
<td>A5</td>
<td>A6</td>
<td>A7</td>
<td>A8</td>
<td>A9</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
<td>B4</td>
<td>B5</td>
<td>B6</td>
<td>B7</td>
<td>B8</td>
</tr>
<tr>
<td>NOT</td>
<td>A1</td>
<td>A2</td>
<td>A3</td>
<td>A4</td>
<td>A5</td>
<td>A6</td>
<td>A7</td>
<td>A8</td>
<td>A9</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
<td>B4</td>
<td>B5</td>
<td>B6</td>
<td>B7</td>
<td>B8</td>
</tr>
</tbody>
</table>

Table 5. Experimental Results of ROOFS and PROOFS.

<table>
<thead>
<tr>
<th>events</th>
<th>ROOFS</th>
<th>PROOFS</th>
<th>ratio</th>
<th>time on 3/60</th>
<th>ROOFS</th>
<th>PROOFS</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>c432</td>
<td>34709</td>
<td>24990</td>
<td>0.72</td>
<td>7.13</td>
<td>5.25</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>c499</td>
<td>33314</td>
<td>17608</td>
<td>0.52</td>
<td>5.82</td>
<td>3.65</td>
<td>0.70</td>
<td></td>
</tr>
<tr>
<td>c880</td>
<td>50735</td>
<td>41484</td>
<td>0.80</td>
<td>11.07</td>
<td>8.52</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>c1355</td>
<td>102320</td>
<td>58647</td>
<td>0.53</td>
<td>23.63</td>
<td>18.03</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>c1908</td>
<td>344285</td>
<td>175426</td>
<td>0.49</td>
<td>61.40</td>
<td>35.10</td>
<td>0.63</td>
<td></td>
</tr>
<tr>
<td>c2670</td>
<td>292599</td>
<td>242314</td>
<td>0.77</td>
<td>67.95</td>
<td>50.60</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>c3540</td>
<td>534731</td>
<td>397315</td>
<td>0.70</td>
<td>99.15</td>
<td>63.67</td>
<td>0.72</td>
<td></td>
</tr>
<tr>
<td>c5315</td>
<td>871832</td>
<td>714157</td>
<td>0.73</td>
<td>189.12</td>
<td>126.80</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>c6288</td>
<td>817484</td>
<td>361118</td>
<td>0.38</td>
<td>116.74</td>
<td>42.88</td>
<td>0.36</td>
<td></td>
</tr>
<tr>
<td>c7552</td>
<td>1362453</td>
<td>1151118</td>
<td>0.79</td>
<td>282.02</td>
<td>182.17</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>s298</td>
<td>155745</td>
<td>67801</td>
<td>0.44</td>
<td>25.77</td>
<td>11.45</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>s349</td>
<td>101116</td>
<td>35086</td>
<td>0.35</td>
<td>15.98</td>
<td>5.67</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>s526</td>
<td>2902862</td>
<td>966891</td>
<td>0.33</td>
<td>456.12</td>
<td>166.25</td>
<td>0.36</td>
<td></td>
</tr>
<tr>
<td>s713</td>
<td>83017</td>
<td>55680</td>
<td>0.67</td>
<td>16.22</td>
<td>9.68</td>
<td>0.60</td>
<td></td>
</tr>
<tr>
<td>s832</td>
<td>590202</td>
<td>414938</td>
<td>0.81</td>
<td>96.72</td>
<td>78.28</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>s838</td>
<td>665868</td>
<td>290877</td>
<td>0.44</td>
<td>108.82</td>
<td>48.73</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>s1238</td>
<td>243096</td>
<td>214400</td>
<td>0.88</td>
<td>60.95</td>
<td>49.87</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>s1494</td>
<td>1770375</td>
<td>1271362</td>
<td>0.72</td>
<td>283.40</td>
<td>168.43</td>
<td>0.59</td>
<td></td>
</tr>
<tr>
<td>s5378</td>
<td>3188203</td>
<td>2321075</td>
<td>0.73</td>
<td>619.07</td>
<td>421.52</td>
<td>0.68</td>
<td></td>
</tr>
<tr>
<td>s35932</td>
<td>11428644</td>
<td>7054431</td>
<td>0.62</td>
<td>1999.20</td>
<td>1356.55</td>
<td>0.68</td>
<td></td>
</tr>
</tbody>
</table>


