Fault Coverage by Pseudo-Random Test Vector Generation Using Combinational Circuits

Authors: *Imran Babar, Arsalan Babar, Syed Khalid*
Department of Electrical and Computer Engineering, University of Illinois, Chicago, IL
Github*: https://github.com/ibabar2/ECE-464-Final-Projectqq*
Mentor: *Professor Wenjing Rao*

1 COMBINATION CIRCUIT REPORT

1.1 *Objective:*

The objective of this study is to find a clever Pseudo Random Test Vector Generation algorithm to get high fault coverage percentage for the faults that are present in a circuit we are testing(C432.bench).

1.2 *Hypothesis:*

If we put taps on 3 taps on alternate bits rather than consecutive bits, then the fault coverage of circuit being tested will be improved. EX:(taps on 1, 3, and 5 rather than 2,3, and 4).

1.3 *Design:*

To test the hypothesis, we decided to edit our project 2 TV set D and TV set E LFSR generation which initially had taps on bits 2, 3, and 4(consecutive) to have taps now at bits 1, 3, and 5(alternative bits). We decided to use same amount of taps as original because adding more taps would mean spending more money and we would not want to do that. Thus, we have altered the design of LFSR to have taps on alternative bits to test whether fault coverage would get improved. To confirm results, we tested this approach on two different circuit benchmarks which obviously generated different fault lists. Furthermore, we have created gnu plots comparing the fault coverage of original results of project 2 which had 3 consecutive taps and the edited version which has taps on alternative bits(1, 3, 5).

1.4 *Data:*

Circuit Benchmark Tested: C432.bench
D(ALT:) generates multiple 8-bit LFSRs with taps at 1, 3, 5 with seeding mechanism being one seed for all (SOLID BLUE LINE).
E(ALT:) Also generates multiple 8-bit LFSRs with taps at 1, 3, 5 with seeding mechanism where s0 is set by user,and s[i+1] is the next vector in the sequence after s[i] (SOLID RED LINE).
D: generates multiple 8-bit LFSRs with taps at 2, 3, 4 with seeding mechanism being one seed for all (SOLID GREEN LINE).
E: Also generates multiple 8-bit LFSRs with taps at 2, 3, 4 with seeding mechanism where s0 is set by user,and s[i+1] is the next vector in the sequence after s[i] (SOLID BLACK LINE).

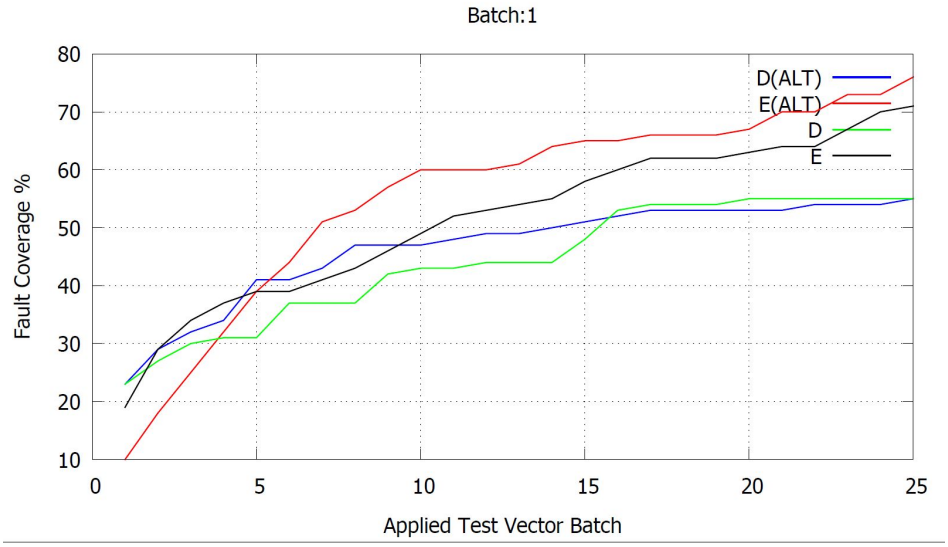

Figure 1. Observer can see that E(ALT) clearly covers a lot more faults than E throughout all applied test vector batch. Meanwhile, D(ALT) at initial applied test vector batch has more fault coverage than D but we see D taking the lead at the very end.

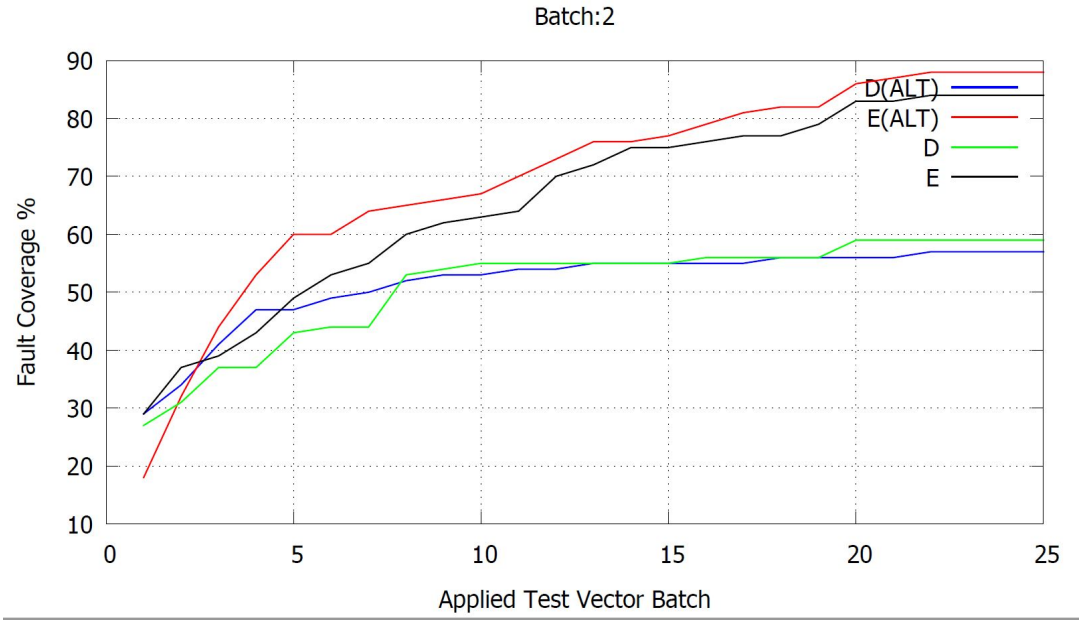

Figure 2. One can note that E(ALT) covers alot more faults than E. Whereas D(ALT) and D stay pretty close when comparing Fault Coverage between the two. Even though D has a better fault coverage than D(ALT), we can see improved performance of D(ALT) from batch 1 to batch 2.

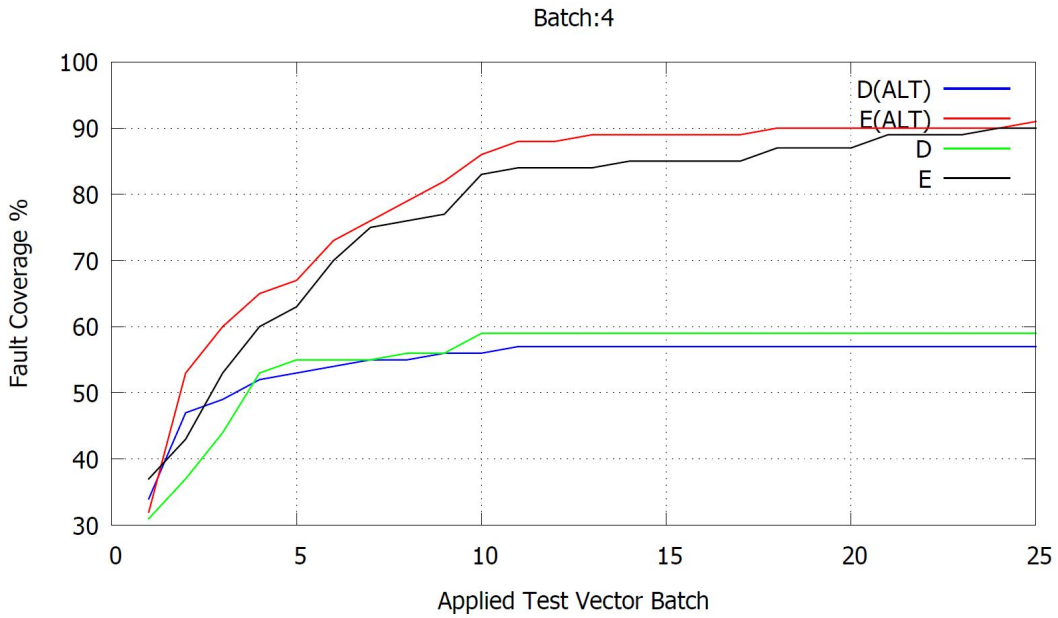

Figure 3. Observers can once again note that E(ALT) covers a lot more faults than E. Whereas D(ALT) and D stay pretty consistent and close to each with D still performing slightly better than D(ALT). However, D(ALT) has almost caught up with D in comparison with previous batches.

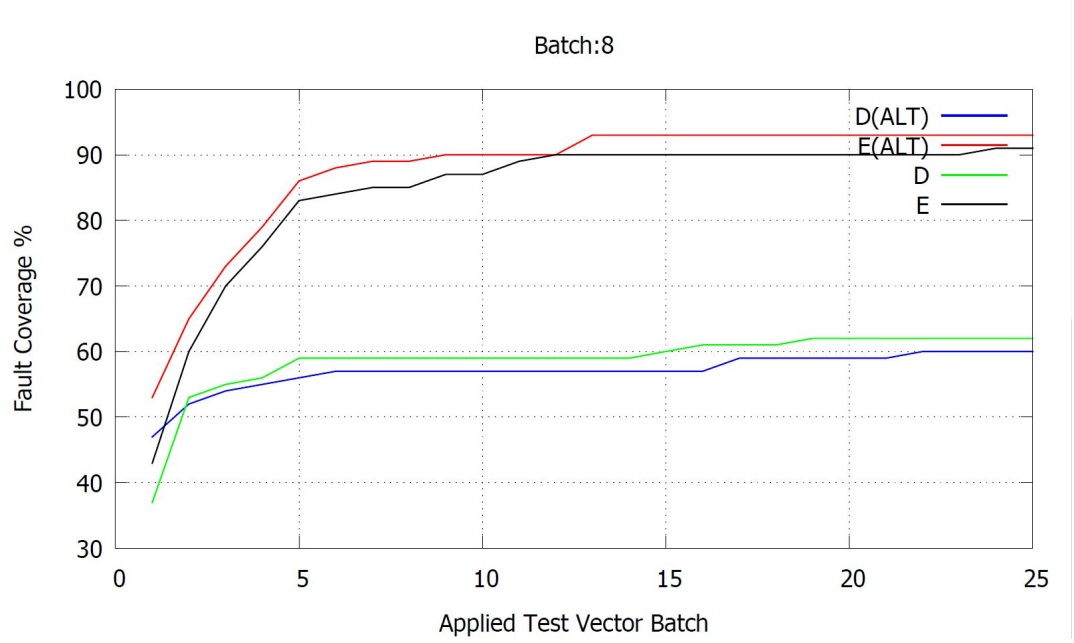

Figure 4. Once again, there is sufficient evidence that E(ALT) produces a lot better fault coverage than E. Meanwhile we see the performance ratio between D(ALT) and D continue to decrease with D still outperforming D(ALT) by slightest of the margins.

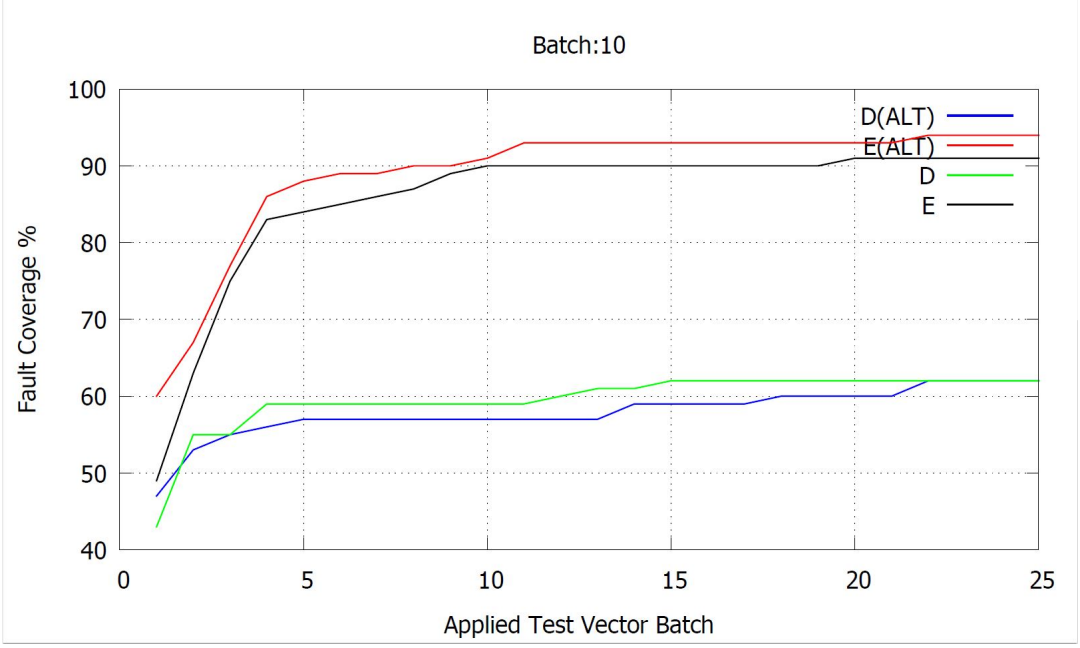

Figure 5. In the final batch, we again see that E(Alt) outperforms E throughout all applied test vector batches. Another interesting thing the observer can see is that, fault coverage performance between D and D(Alt) levels at the very end. D(Alt) continued to improve its performance and has finally leveled in fault coverage performance with D.

1.5 *Conclusion:*

After closely analyzing the graphs above, it can clearly be concluded that E(ALT) has a better fault coverage performance than E throughout the testing and that D(ALT) has improved its fault coverage performance throughout and possibly for higher batches it will outperform D. Firstly, E(Alt) has dominated and has never let E outperform. Furthermore, we have the performance ratio of E(Alt) increasing in comparison to E. Next, we have seen D dominating D(alt) by slightest of margins, however we have seen D(Alt) improving its fault coverage performance with higher batches. For instance, at Batch 10, from applied test vector batches 22-25, we see D(Alt) leveling fault coverage performance with D. As Batch size has increased we have the performance ratio of D(alt) improving when compared to D. Thus, it can safely be said, that If we put taps on alternating bits rather than consecutive bits, then the fault coverage of circuit being tested will be improved.

2 SEQUENTIAL CIRCUIT REPORT

2.1 *Objective:*

To get highest fault coverage percentage possible by looking at Flip Flop initialization pattern.

2.2 *Hypothesis:*

If we initialize the flip flops to either 0 or 1, then sequential simulator will produce a higher fault coverage in less cycles.

2.3 *Design:*

To test the hypothesis, we ran the sequential simulator with three different configurations. First, we ran the simulation with flip flops initialized as unknown for a custom fault list and noted how many faults went detected and undetected. Next, we repeated the same experiment with flip flops initialized as 0 and 1 and once again noted how many faults were detected out of the total faults tested.

2.4 ***Data:***

Configuration 1:
Flip Flop: Unknown
Circuit Benchmark: S27.bench
Test Vector: 20
Cycles: 100

Configuration 2:
Flip Flop: 1
Circuit Benchmark: S27.bench
Test Vector: 20
Cycles: 100

Configuration 3:
Flip Flop: 0
Circuit Benchmark: S27.bench
Test Vector: 20
Cycles: 100

Fault Tested	FF as Unknown	FF as 1	FF as 0
G0-SA-0	Undetected	Detected	Detected
G0-SA-1	Detected	Undetected	Undetected
G1-SA-1	Undetected	Detected	Detected
G1-SA-0	Undetected	Detected	Detected
G5-IN-G10-SA-0	Undetected	Undetected	Undetected
G11-SA-0	Detected	Detected	Detected
G5-SA-0	Undetected	Undetected	Undetected
G5-SA-1	Detected	Detected	Detected
G11-IN-G9-SA-0	Undetected	Detected	Detected
G9-IN-G16-SA-1	Undetected	Detected	Detected

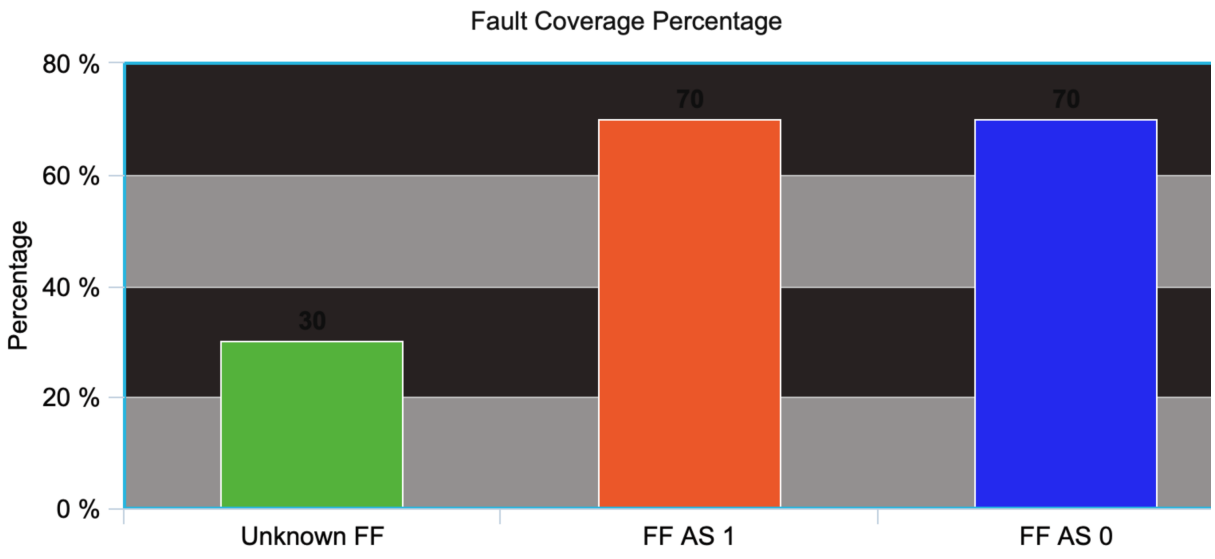

Figure 6. Observer can note that when Flip flops are initialized as 1 or 0, there is clearly higher fault coverage.

2.5 *Conclusion:*

After testing our hypothesis, we came to the conclusion that if we initialize the flip flops as 0 or 1, the probability of having a higher fault coverage is better because initializing flip flops with unknowns at the beginning would require few extra cycles to set flip flops to certain finite value. The results back our conclusion with unknown flip flop detecting only 30 percent of faults meanwhile flip flops that were set as 0 or 1 initially had a fault coverage of 70 percent. To confirm results, we have tested few other benchmarks, and results support our conclusion.