VLSI Design Verification and Testing

Sequential Circuit ATPG

Mohammad Tehranipoor
Electrical and Computer Engineering
University of Connecticut

Overview

- Motivation
- Sequential circuit ATPG
- An example test generation
- Time-frame expansion
  - Nine-valued logic
  - ATPG implementation and drivability
  - Complexity of ATPG
  - Cycle-free and cyclic circuits
- Test generations systems
  - Classification
  - Forward time test generator – FASTEST
  - General comments
  - Simulation based test generators – contest, strategate
- Summary

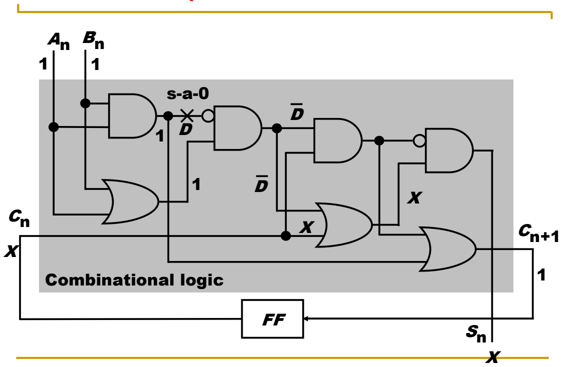
Motivation

- A sequential circuit has memory in addition to combinational logic.
- Test for a fault in a sequential circuit is a sequence of vectors, which
  - Initializes the circuit to a known state
  - Activates the fault, and
  - Propagates the fault effect to a primary output

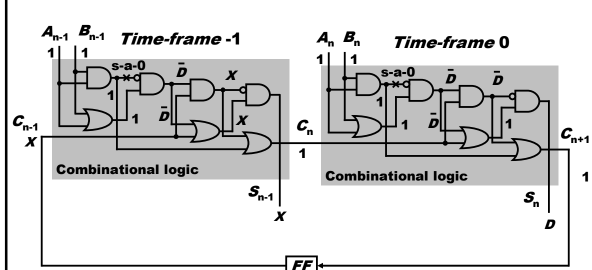
Sequential Circuit ATPG

- Methods
  - Time-frame expansion methods
    - Forward time, reverse time, forward and reverse time
  - Simulation-based methods

Example: A Serial Adder

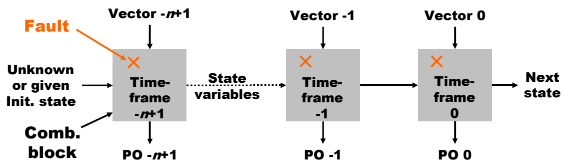


Time-Frame Expansion

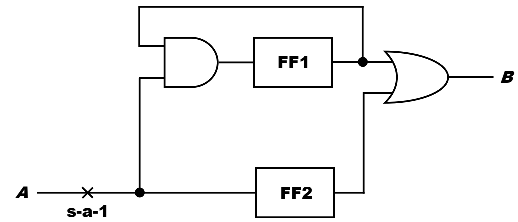


Concept of Time-Frames

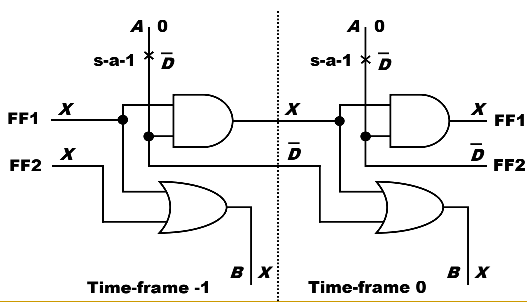
- If the test sequence for a single stuck-at fault contains *n* vectors,
  - Replicate combinational logic block *n* times
  - Place fault in each block
  - Generate a test for the multiple stuck-at fault using combinational ATPG with 9-valued logic



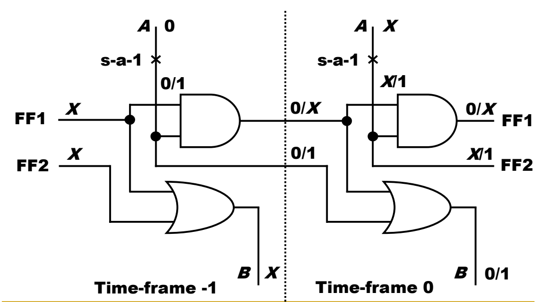
Example for Logic Systems



Five-Valued Logic (Roth)
0,1, *D*, $\overline{D}$, *X*



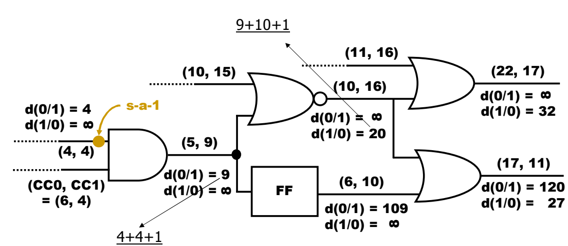
Nine-Valued Logic (Muth)
0,1, 1/0, 0/1, 1/*X*, 0/*X*, *X*/0, *X*/1, *X*



An implementation of ATPG

- Select a PO for fault detection based on drivability analysis.
- Place a logic value, 1/0 or 0/1, depending on fault type.
- Justify the output value from PIs, considering all necessary paths and adding backward time-frames.
- If justification is impossible, then use drivability to select another PO and repeat justification.
- If the procedure fails for all reachable POs, then the fault is *untestable.*
- If 1/0 or 0/1 cannot be justified at any PO, but 1/X or 0/X can be justified, then the fault is *potentially detectable.*

Drivability Example

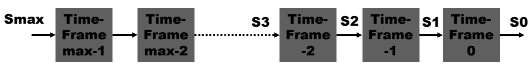


CC0 and CC1 are SCOAP combinational controllabilities

d(0/1) and d(1/0) of a line are effort measures for driving a specific fault effect to that line

Complexity of ATPG

- Synchronous circuit -- All flip-flops controlled by clocks; PI and PO synchronized with clock:
  - Cycle-free circuit – No feedback among flip-flops: Test generation for a fault needs no more than *dseq* + 1 time-frames, where *dseq* is the sequential depth.
  - Cyclic circuit – Contains feedback among flip-flops: May need 9^{Nff} time-frames, where *Nff* is the number of flip-flops.
- Asynchronous circuit – Higher complexity!

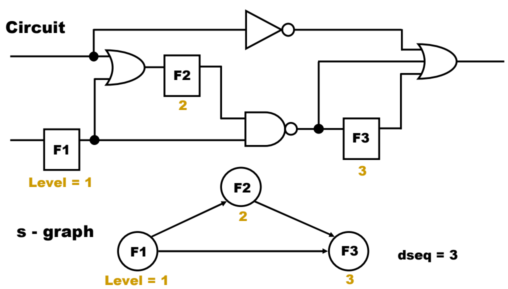


max **= Number of distinct vectors with 9-valued elements =** 9^{Nff}

Cycle-Free Circuits

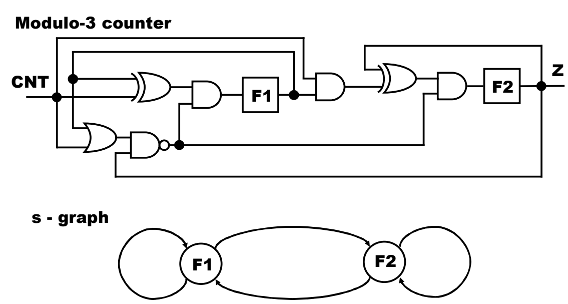
- Characterized by absence of cycles among flip-flops and a sequential depth, *dseq*.
- *dseq* is the maximum number of flip-flops on any path between PI and PO.
- Both good and faulty circuits are initializable.
- Test sequence length for a fault is bounded by *dseq* + 1.

Cycle-Free Example



All faults are testable. See Example 8.6.

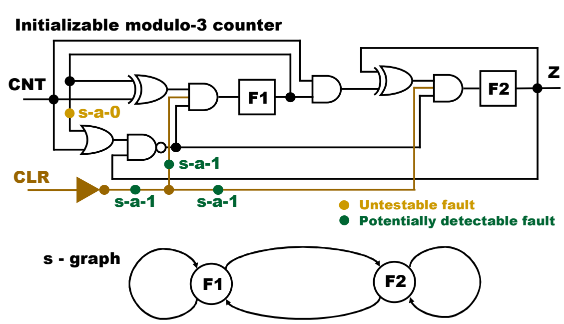
Cyclic Circuit Example



Modulo-3 Counter

- Cyclic structure – Sequential depth is undefined.
- Circuit is not initializable. No tests can be generated for any stuck-at fault.
- After expanding the circuit to 9^{Nff} = 81, or fewer, time-frames ATPG program calls any given target fault untestable.
- Circuit can only be functionally tested by multiple observations.
- Functional tests, when simulated, give no fault coverage.

Adding Initializing Hardware



Benchmark Circuits

Circuit	s1196	s1238	s1488	s1494
PI	14	14	8	8
PO	14	14	19	19
FF	18	18	6	6
Gates	529	508	653	647
Structure	Cycle-free	Cycle-free	Cyclic	Cyclic
Seq. depth	4	4	--	--
Total faults	1242	1355	1486	1506
Detected faults	1239	1283	1384	1379
Potentially detected faults	0	0	2	2
Untestable faults	3	72	26	30
Abandoned faults	0	0	76	97
Fault coverage (%)	99.8	94.7	93.1	91.6
Fault efficiency (%)	100.0	100.0	94.8	93.4
Max. sequence length	3	3	24	28
Total test vectors	313	308	525	559
Gentest CPU s (Sparc 2)	10	15	19941	19183

Test Generations Systems

- Classification
  - Target a fault
    - Reverse-time processing
    - Forward-time processing
    - Forward and reverse-time processing
  - Target no specific fault
    - Simulation based algorithms

Test Generations Systems

- Reverse-time processing
  - Determine a PO where the fault-effect will appear
  - Backtrace within the time frame to excite and or propagate a fault/fault-effect
  - If not possible go add a timeframe (previous timeframe) and continue

Test Generations Systems
Reverse-time processing

Positives and Negatives

Low memory usages

Timeframe added when needed

Ability to determine if a fault is untestable

Hard to determine the PO where fault will be detected

During backward motion, often the timeframe is assumed to be fault-free, this can generate invalid tests

Test application is in the order opposite to test generation

Test Generations Systems

- Forward-time processing
  - Excite a fault in the present timeframe
  - If excited, propagate to an output, else add a timeframe and then excite – continue till fault excited
  - Try to propagate the fault, if not successful, add timeframe and continue the process till fault detected at a PO

Test generations systems
Forward-time processing

FASTEST approach

- Use controllability values to determine the timeframe where the fault can be excited
- Use observability values to determine the timeframe where the fault will be observed
- Together these will determine the number of timeframes need to detect the fault of interest
- Work with that many timeframes in combinational mode to generate a test sequence in forward time

Test generations systems

- Forward and reverse-time processing
  - Perform the fault effect propagation in forward time
  - Perform excitation (justification) in reverse time using fault-free circuit

Test Generations Systems
Forward and reverse-time processing

Positives and Negatives

Medium memory usages

Timeframe added when needed in reverse as well as in forward time

Ability to determine if a fault is untestable

During backward motion, often the timeframe is assumed to be fault-free, this can generate invalid tests

Test application is in the order opposite to test generation

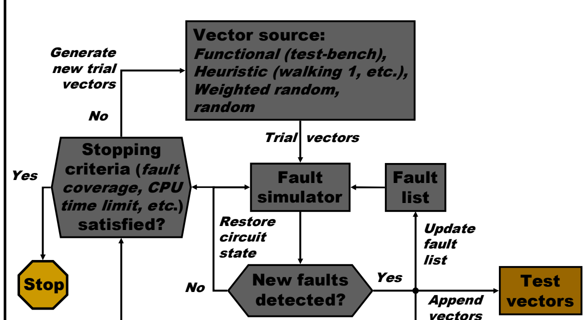
Test Generations Systems

- General comments
  - Store state information during test generation for later use
  - Preprocess the circuit and learn about implication etc.
  - Reuse previous solutions
  - Modify easy/hard and SCOAP to better suit needs of the sequential ATPGs
  - Make a better selection of the target fault – as in FASTEST
  - Neither 5-v nor 9-v are complete algorithms for sequential ATPG
  - Multiple timeframe observation a possible solution but has not found way in practice

Simulation based systems

- Difficulties with time-frame method:
  - Long initialization sequence
  - Impossible initialization with three-valued logic (Section 5.3.4)
  - Circuit modeling limitations
  - Timing problems – tests can cause races/hazards
  - High complexity
  - Inadequacy for asynchronous circuits
- Advantages of simulation-based methods
  - Advanced fault simulation technology
  - Accurate simulation model exists for verification
  - Variety of tests – functional, heuristic, random
  - Used since early 1960s

Using Fault Simulator



Contest

- A Concurrent test generator for sequential circuit testing (Contest).
- Search for tests is guided by cost-functions.
- Three-phase test generation:
  - Initialization – no faults targeted; cost-function computed by true-value simulator.
  - Concurrent phase – all faults targeted; cost function computed by a concurrent fault simulator.
  - Single fault phase – faults targeted one at a time; cost function computed by true-value simulation and dynamic testability analysis.
- Ref.: Agrawal, *et al.*, IEEE-TCAD, 1989.

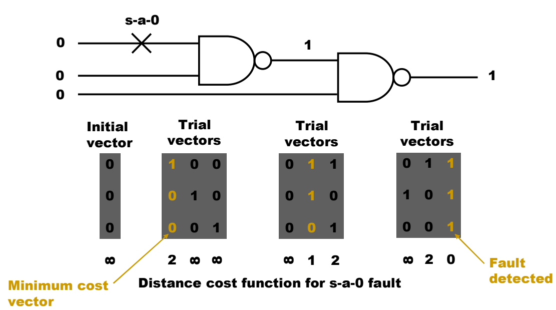
Phase I: Initialization

- Initialize test sequence with arbitrary, random, or given vector or sequence of vectors.
- Set all flip-flops in *unknown* (*X*) state.
- Cost function:
  - Cost = Number of flip-flops in the unknown state
  - Cost computed from true-value simulation of trial vectors
- Trial vectors: A heuristically generated vector set from the previous vector(s) in the test sequence; e.g., all vectors at unit Hamming distance from the last vector may form a trial vector set.
- Vector selection: Add the minimum cost trial vector to the test sequence. Repeat trial vector generation and vector selection until cost becomes zero or drops below some given value.

Phase II: Concurrent Fault Detection

- Initially test sequence contains vectors from Phase I.
- Simulate all faults and drop detected faults.
- Compute a *distance cost function* for trial vectors:
  - Simulate all undetected faults for the trial vector.
  - For each fault, find the shortest *fault distance* (in number of gates) between its fault effect and a PO.
  - Cost function is the sum of fault distances for all undetected faults.
- Trial vectors: Generate trial vectors using the unit Hamming distance or any other heuristic.
- Vector selection:
  - Add the trial vector with the minimum distance cost function to test sequence.
  - Remove faults with zero fault distance from the fault list.
  - Repeat trial vector generation and vector selection until fault list is reduced to given size.

Distance Cost Function



Phase III: Single Fault Target

- Cost (fault, input vector) = *K* x AC + PC
  - *Activation cost* (AC) is the dynamic controllability of the faulty line.
  - *Propagation cost* (PC) is the minimum (over all paths to POs) dynamic observability of the faulty line.
  - *K* is a large weighting factor, e.g., *K* = 100.
  - Dynamic testability measures (controllability and observability) are specific to the present signal values in the circuit.
  - Cost of a vector is computed for a fault from true-value simulation result.
  - Cost = 0 means fault is detected.
- Trial vector generation and vector selection are similar to other phases.

Contest Result: s5378+

- 35 PIs, 49 POs, 179 FFs, 4,603 faults.
- Synchronous, single clock.

	Contest	Random vectors	Gentest**
Fault coverage	75.5%	67.6%	72.6%
Untestable faults	0	0	122
Test vectors	1,722	57,532	490
Trial vectors used	57,532	--	--
Test gen. CPU time#	3 min.*	0	4.5 hrs.
Fault sim. CPU time#	9 min.*	9 min.	10 sec.

+ Results provided by the developers of Contest
# Sun Ultra II, 200MHz CPU *Estimated time
**Time-frame expansion (higher coverage possible with more CPU time)

Genetic Algorithms (GAs)

- Theory of evolution by natural selection (Darwin, 1809-82.)
  - C. R. Darwin, *On the Origin of Species by Means of Natural Selection*, London: John Murray, 1859.
  - J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.
  - D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning, Reading*, Massachusetts: Addison-Wesley, 1989.
  - P. Mazumder and E. M. Rudnick, *Genetic Algorithms for VLSI Design, Layout and Test Automation*, Upper Saddle River, New Jersey, Prentice Hall PTR, 1999.
- Basic Idea: Population *improves* with each generation.
  - Population
  - Fitness criteria
  - Regeneration rules

GAs for Test Generation

- Population: A set of input vectors or vector sequences.
- Fitness function: Quantitative measures of population succeeding in tasks like initialization and fault detection (reciprocal to cost functions.)
- Regeneration rules (heuristics): Members with higher fitness function values are *selected* to produce new members via transformations like *mutation* and *crossover*.

Strategate Results

	s1423	s5378	s35932
Total faults	1,515	4,603	39,094
Detected faults	1,414	3,639	35,100
Fault coverage	93.3%	79.1%	89.8%
Test vectors	3,943	11,571	257
CPU time HP J200 256MB	1.3 hrs.	37.8 hrs.	10.2 hrs.

Ref.: M. S. Hsiao, E. M. Rudnick and J. H. Patel, "Dynamic State Traversal for Sequential Circuit Test Generation," *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 5, no. 3, July 2000.

Summary

- Combinational ATPG algorithms are extended:
  - Time-frame expansion unrolls time as combinational array
  - Nine-valued logic system
  - Justification via backward time
- Cycle-free circuits:
  - Require at most *dseq* time-frames
  - Always initializable
- Cyclic circuits:
  - May need 9^{Nff} time-frames
  - Circuit must be initializable
  - Partial scan can make circuit cycle-free (Chapter 14)

Summary (contd.)

- Sequential test generators classified
- FASTEST discussed
- Fault simulation is an effective tool for sequential circuit ATPG.
- Simulation-based methods produce more vectors, which can potentially be reduced by compaction.
- A simulation-based method and purely foreward time test generators cannot identify untestable faults.

History of simulation based TGs

- Seshu and Freeman, 1962, Asynchronous circuits, parallel fault simulator, single-input changes vectors.
- Breuer, 1971, Random sequences, sequential circuits
- Agrawal and Agrawal, 1972, Random vectors followed by D-algorithm, combinational circuits.
- Shuler, *et al.*, 1975, Concurrent fault simulator, random vectors, sequential circuits.
- Parker, 1976, Adaptive random vectors, combinational circuits.
- Agrawal, Cheng and Agrawal, 1989, Directed search with cost-function, concurrent fault simulator, sequential circuits.
- Srinivas and Patnaik, 1993, Genetic algorithms; Saab, *et al.*, 1996; Corno, *et al.*, 1996; Rudnick, *et al.*, 1997; Hsiao, *et al.*, 1997.