Sequential Circuit Testing 3

Recap: Approaches

- State table analysis
- Machine identification (checking sequence) method
- Time-frame expansion

Misc. Issues

- Controlling and observing internal states of a sequential circuit

  Scan design solves this problem
- Establishing a known initial state

  Need for backtrace through multiple time-frames
- Need for nine-valued logic
- Easy special case: cycle-free circuits

Nine-Valued Logic

- For sequential testing we need may to consider all pairs of 0,1,X signals for the good/faulty machines
- Roth's 5-valued algebra:

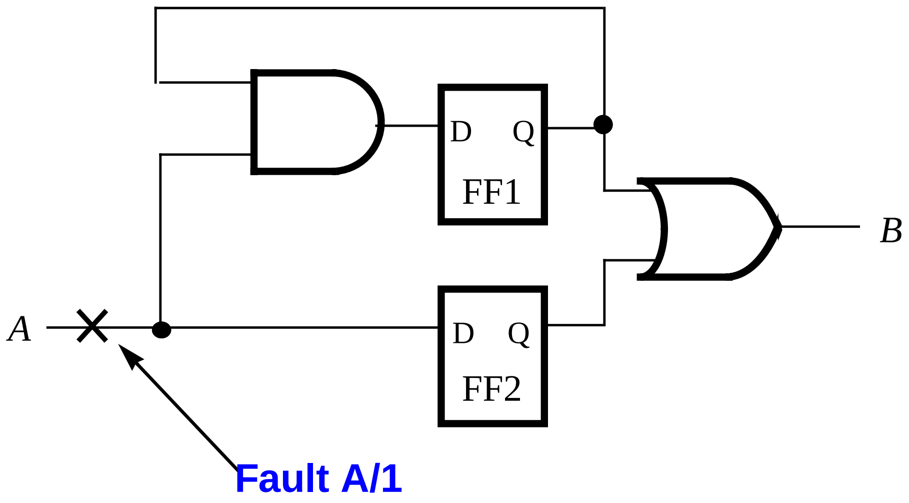
D	**D′**	**0**	**1**	**X**
1/0	0/1	0/0	1/1	X/X

- Muth's 9-valued algebra:

D	**D′**	**0**	**1**	**X**				
1/0	0/1	0/0	1/1	X/X	0/X	1/X	X/0	X/1

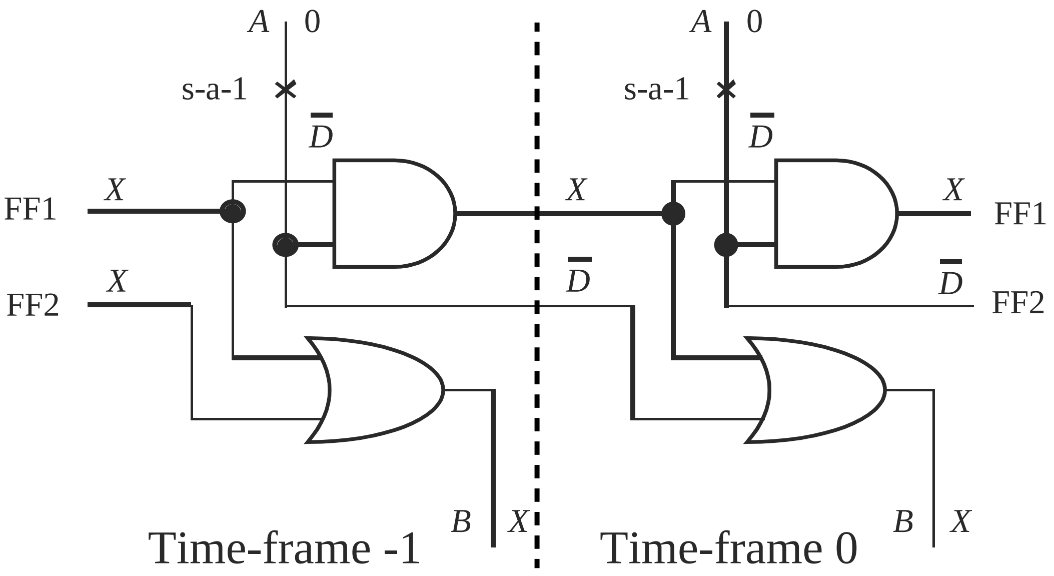
Nine-Valued Logic

Example: Need for 9 values



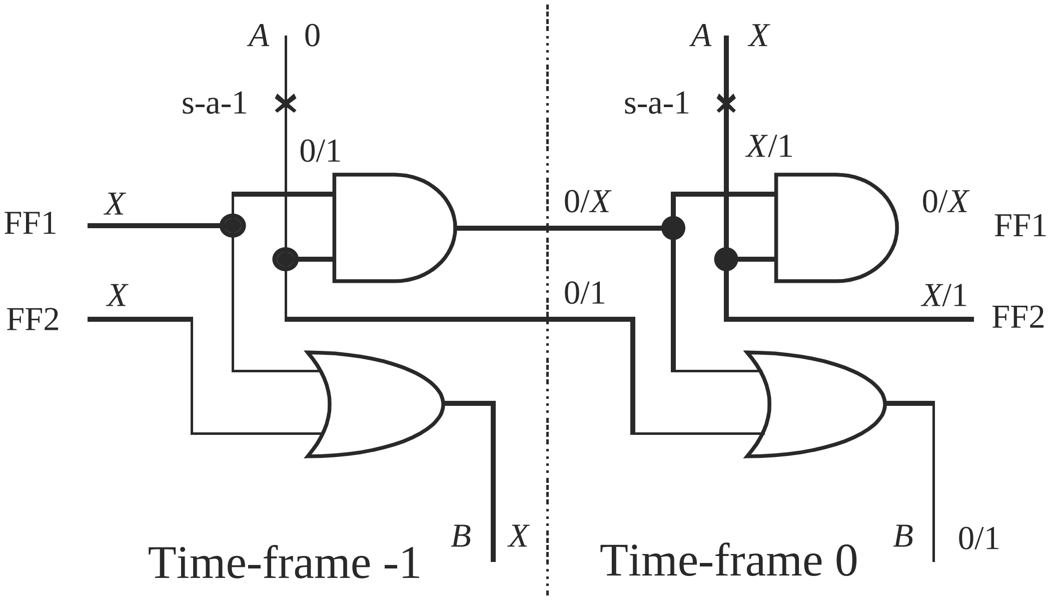
Nine-Valued Logic

Example: ATPG with 5 values



Nine-Valued Logic

Example: ATPG with 9 values



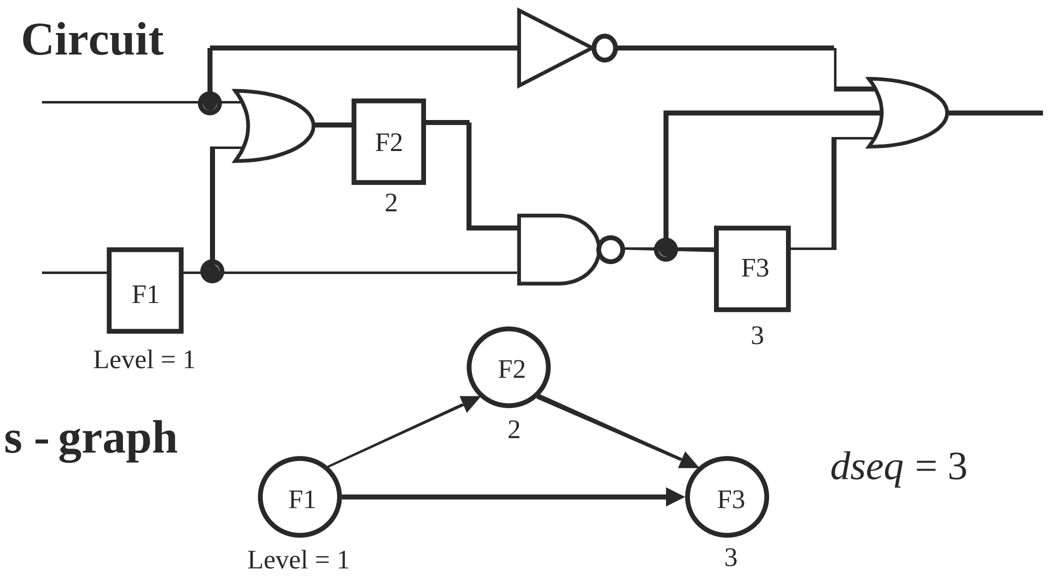
Cycle-Free Circuits

- 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 PIs and POs
- Special testing properties

  Both good and faulty circuits are initializable

  Test sequence length for a fault is bounded by *dseq* + 1

Cycle-Free Circuits



dseq = 3

Simulation-Based ATPG

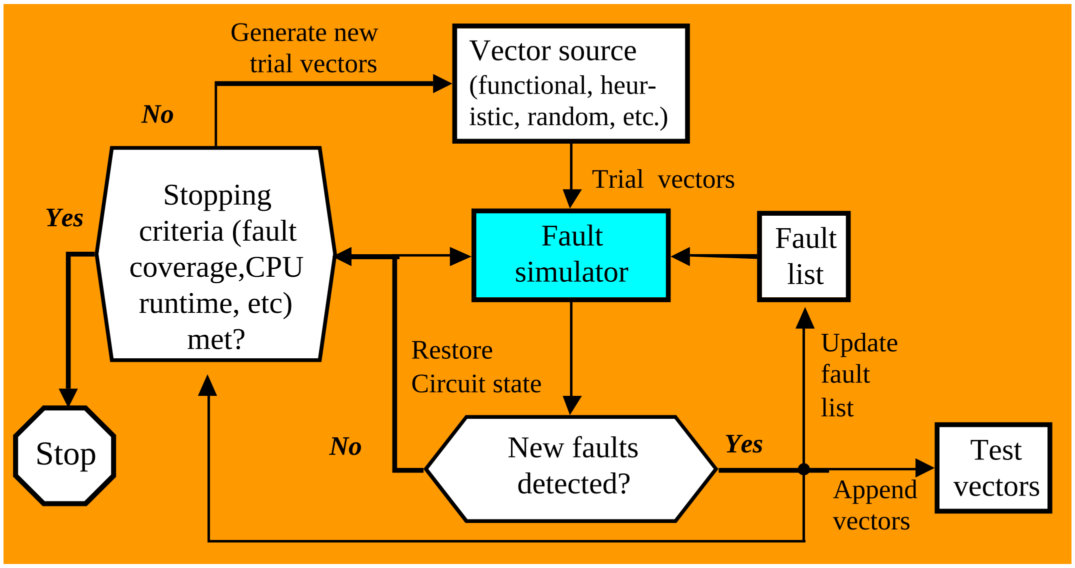
Possible Methods

- Directed search
- Genetic algorithms
- Spectral techniques

Advantages

- Fault simulation technology is very well-developed
- Many types of test methods (deterministic, functional, random) can be used

Simulation-Based ATPG



CONTEST Algorithm

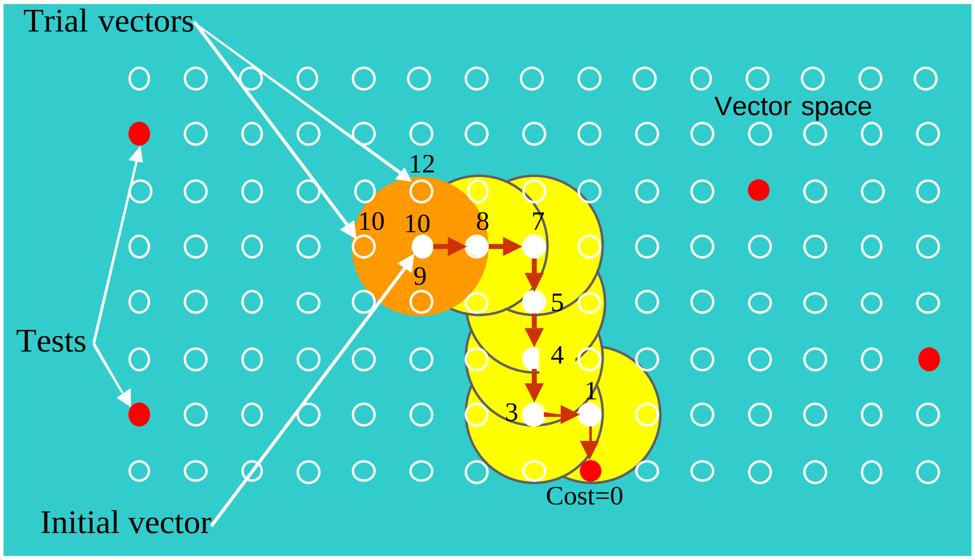
- Simulation-based ATPG program for sequential circuits developed at Bell Labs (1989)
- Searches for tests guided by cost functions
- Uses a 3-phase test generation method:

  Initialization phase: No faults are targeted. Cost function is computed by true-value simulation.

  Concurrent phase: All faults are targeted. Cost function is computed by concurrent fault simulation.

  Single fault phase: Faults are targeted one at a time. Cost function is computed by true-value simulation and dynamic testability analysis

CONTEST Algorithm



CONTEST Algorithm

Cost Functions

- Functions are defined for specific current objectives (initialization or fault detection)
- Each function numerically grades a test vector for suitability to meet the current objective
- Cost function = 0 for any vector that exactly meets the objective
- Cost is computed for an input vector via either true-value or fault simulation

CONTEST Algorithm

Phase I: Initialization

- Initialize test sequence with random or given vector (sequence)
- Set all flip-flops in unknown (X) state
- Cost functions used:
    - Number of flip-flops in the X state
    - Cost computed from true-value simulation of trial vectors
- Trial vector generation: Heuristically generate trial vector set from the previous vector(s) in the test sequence, e.g., all vectors at Hamming distance one from the last vector
- Vector selection: Add the minimum-cost trial vector to the test sequence. Repeat trial vector generation and vector selection until cost ≤ given limit.

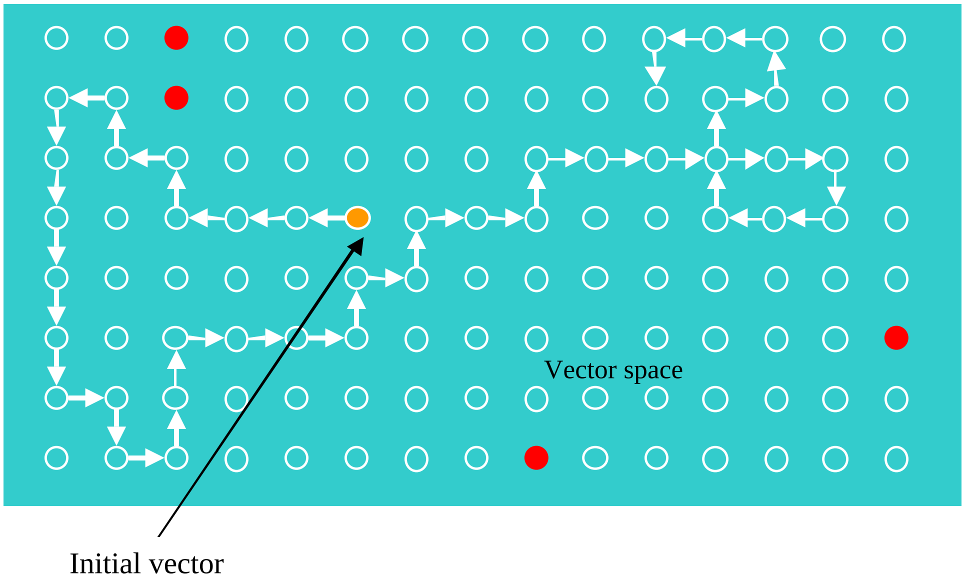
CONTEST Algorithm

Phase II: Concurrent Fault Detection

- Initial test sequence uses vectors from Phase I
- Simulate all faults and drop detected faults
- Compute a distance cost function for trial vectors:
    - For each undetected fault, find the shortest fault distance
        (no. of gates) between its fault effect and a primary output
    - Cost function = sum of fault distances for all undetected faults
- Trial vectors: Generate trial vectors, e.g., using unit distance
- Vector selection:
    - Add trial vector with the minimum 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 reduces to given size

CONTEST Algorithm

Need for Phase III



CONTEST Algorithm

Phase III: Single Fault Target

- Cost (fault, input vector) = K × 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 the true-value simulation result.

  Cost = 0 means fault is detected.

- Trial vector generation and vector selection are similar to the other phases

CONTEST Algorithm

Test Results: Benchmark circuit: 35 PIs, 49 POs, 179 flip-flops, 4,603 faults, synchronous

	CONTEST	Random Tests	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.

# Sun Ultra II, 200 MHz CPU *Estimated time
**Time-frame expansion (higher coverage possible with more CPU time)

Genetic Algorithm

- Heuristic algorithms that mimic the biological mechanisms underlying the theory of evolution by natural selection (Darwin 1859)
- *Key idea*: Populations improve with each generation.
- Genetic ATPG algorithms

  Population: Set of input vectors or vector sequences.

  Fitness function: Quantitative measures of testing success in tasks like initialization and fault detection

  Regeneration rules: Tests with higher fitness values are selected to produce new tests via transformations like "mutation" and "crossover."

High-Level Test Generation

Goals

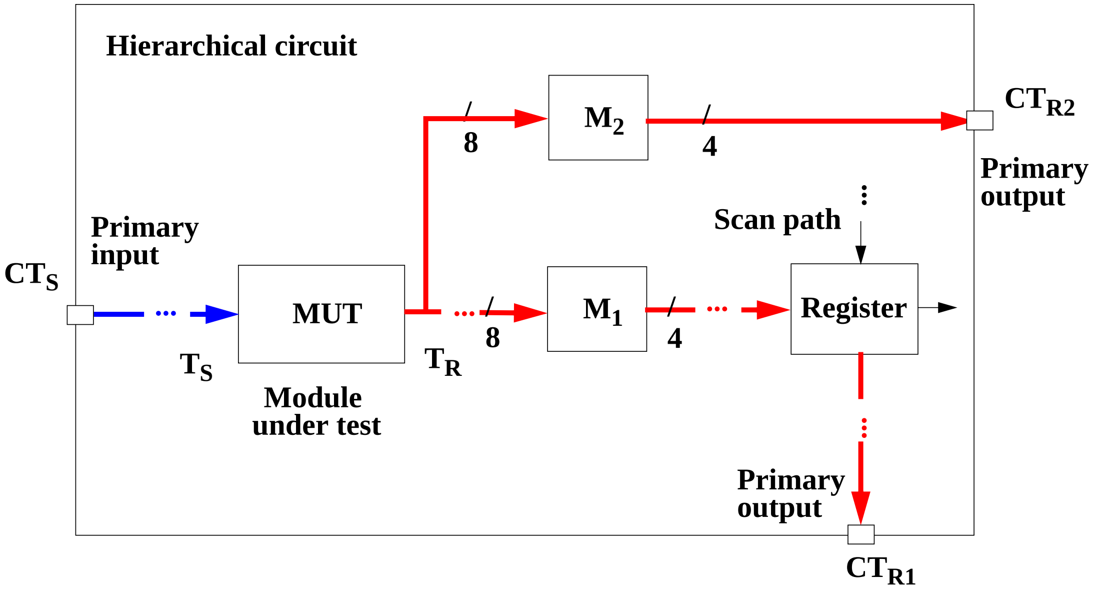
- Speed up test generation
- Generate tests for circuits without complete structural models

Methods

- Use structural hierarchy and circuit and fault modeling
- Use functional descriptions of modules and circuits
- Distinguish data and control functions
- Exploit high-level (expert) information about circuit operation

High-Level Test Generation

Precomputed Tests for Modules


Hierarchical circuit
M_2
8
4
CT_{R2}
Primary output
Scan path
Primary input
CT_S
MUT
T_R
8
M_1
4
Register
T_S
Module under test
Primary output
CT_{R1}