**EECS 579: Test Generation 4**

**Other Combinational ATPG Algorithms**

**SOCRATES**

- Structure-Oriented Cost-Reducing Automatic Test pattern generation [Schultz et al. 1988]
- An ATPG “system”—not just a test generation algorithm
- **Main Features**
  - “Learning-based” ATPG
  - Seeding with random tests
  - Fast combinational fault simulation built-in
  - More complex implications (= learning)
  - More complex unique sensitization
  - Allows higher-level primitives

---

**Test Generation System**

- **Compactor** → **Circuit description** → **SOCRATES with fault simulator** → **Fault list**
  - **Aborted faults**
  - **Undetected faults**
  - **Redundant faults**
  - **Backtrack distribution**
**Learning Strategy**

1. Contrapositive law: \((P \Rightarrow Q)\) is equivalent to \((\lnot Q \Rightarrow \lnot P)\)
2. \((a = 1) \Rightarrow (f = 1)\), so \((f = 1) \Rightarrow (a = 1)\), so \((f = 0) \Rightarrow (a = 0)\)

**Unique Sensitization**

- When \(a\) is the only D-frontier signal, find dominators of \(a\) and set their inputs unreachable from \(a\) to 1
- Find dominators of single D-frontier signal \(a\) and make their common input signals non-controlling
**Critical Path ATPG Method**

- References: Abramovici et al. book [1990]. Originally used in LASAR (Logic Automated Stimulus And Response) [Thomas 1971]

- **Main Features**
  - “Fault-independent” ATPG
  - Derives tests for many faults without targeting specific faults
  - Relies primarily on Justification procedure
  - Values assigned to lines on sensitized paths are termed “critical”

**Basic Steps**

- Select a PO and assign it a (critical) 0 or 1 value $v$
- Recursively justify $v$ by assigning critical values to gate inputs wherever possible

---

![Critical Path Method Diagram]

- Critical 0
- Critical 1
- Non-critical 1
### Critical Path Method

**Advantages**
- Eliminates need for separate fault simulation step
- New tests can be quickly generated by modifying preceding tests

**Disadvantages**
- Test generation steps tend to overlap causing unnecessary repetition
- Conflicts occur when justifying several paths. May have to replace critical patterns (cubes) by noncritical patterns
- Does relatively unsystematic exploration of circuit paths
- Multiple-path sensitization is usually ignored, so some detectable faults may be missed

```plaintext
CPTGFF()
begin
while (Critical # Ø)
begin
  remove one entry (i,val) from Critical
  set i to val
  mark i as critical
  if j is a gate output then
  begin
    c = controlling value of i
    i = inversion of i
    inval = val \& i
    if inval = 0
    then for every input j of i
      add (c,j) to Critical
    else
      begin
        for every input j of i
        begin
          add (i,c) to Critical
          for every input k of i other than j
          Justify (k,c)
          CPTGFF()
        end
      end
  end
end
j* Critical = Ø
* record new test
return
end
```
**Random Test Generation**

- Uses a simple ATPG algorithm implemented in hardware or software
- Random actually means pseudorandom
  
  Resembles a random in that every pattern is equally likely
  (uniform distribution)
  
  Test sequences are repeatable (and technically deterministic)
  
  Test vectors are not usually repeated
  
  Simulation is required to determine fault coverage
- May be combined with deterministic testing in hybrid approach
  1. Use random testing to detect “easy” faults
  2. Use deterministic ATPG algorithm (PODEM, FAN, etc.) to
detect the residual “hard” faults
- Generally requires far more test vectors to achieve high coverage

**Quality Measures**

- **Testing quality** $T_N$: Probability that all detectable SSL faults are
detected by $N$ random vectors

- **Detection quality** $d_N$: Probability that the hardest-to-detect SSL fault
is detected by $N$ random vectors. $d_N = \min d_N^f$ over all faults \{f\}

- If $T_f$ is the set of all faults that detect fault $f$ in an n-input circuit, then
the probability that a random vector detects $f$ is

$$d_f = |T_f|/2^n$$

- Estimate of the number $N$ of random tests needed to achieve testing
quality $c$ [Savir and Bardell 1984]:

$$N_f = \left\lceil \frac{\ln(1-c) - \ln(k)}{\ln(1-d_{\text{min}})} \right\rceil$$

where $k$ is the number of faults with detection probability between
$d_{\text{min}}$ and $2d_{\text{min}}$
**General ATPG Quality Measures**

- Fault Coverage = No. of detected faults / Total no. of faults
- Fault Efficiency = No. of detected faults / (Total no. of faults – No. of undetectable faults)
- Test generation time = (Normalized) CPU time to generate tests for all detectable faults on benchmark circuits

**Benchmark Circuits**
ISCAS-85 (combinational) and ISCAS-89 (sequential)

www.cbl.ncsu.edu/CBL_Docs/Bench.html#BenchList

**Test Set Compaction**

**Simulation-Based Compaction**

- Suppose a test set with deterministic and random patterns has been generated
- Fault-simulate the test patterns in reverse order of generation
  - Examine the deterministic patterns first
  - Examine the randomly-generated patterns last
- When coverage reaches 100%, discard all remaining test patterns
- This usually significantly reduces the test set
**Test Set Compaction**

**Static Compaction**
- ATPG algorithm should leave unassigned inputs as X
- Two patterns are *compatible* if they have no conflicting values for any primary input
- Combine two compatible tests \( t_a \) and \( t_b \) into one test \( t_{ab} \) using D-intersection
- The combined tests \( t_{ab} \) detects the union of faults detected by \( t_a \) and \( t_b \)
- After ATPG, combine as many tests as possible (order-dependent)

**Dynamic Compaction**
- Process every partially-assigned ATPG pattern immediately
- Assign 0 or 1 to unassigned (X) primary inputs in a way that detects additional faults

**Example**
- Given the four member test set \{t1,t2,t3,t4\}, where
  \[
  \begin{align*}
  t_1 &= 0 \ 1 \ X \\
  t_2 &= 0 \ X \ 1 \\
  t_3 &= 0 \ X \ 0 \\
  t_4 &= X \ 0 \ 1 
  \end{align*}
  \]
- Combine \( t_1 \) and \( t_3 \) first. Then combine \( t_2 \) and \( t_4 \) to obtain:
  \[
  \begin{align*}
  t_{13} &= 0 \ 1 \ 0 \\
  t_{24} &= 0 \ 0 \ 1 
  \end{align*}
  \]
- Test set size has been reduced from reduced from 4 to 2
Sequential Circuit Testing

Characteristics

- Tests are sequences of input vectors
- Much harder to test than combinational circuits
- Two circuit types: asynchronous and synchronous

Basic Approaches

- State table analysis
- Machine identification (checking sequence) method
- Time-frame expansion: combinational ATPG methods such as PODEM) are applied to an iterative logic array (ILA) model of the target sequential circuit
- Simulation-based methods

Key Assumptions

- Unknown initial state (self-initializing tests)
- Known initial state