Sequential Circuits

- **Approach**
  - Convert Finite State Machine to Corresponding Iterative Network
    - Multiple Time Frames (Iterative Cells) Needed for
    - Justification and Propagation
  - One Fault in Sequential Circuit
    - Many Faults in Corresponding Iterative Network
    - Use 9-valued signals

- **Issues**
  - Order of Justification and Propagation
  - Simulation Values
  - Test Point Insertion (Partial Scan)
Sequential ATPG

- Difficulties
  - Initialization of the bistables
  - Gated clocks
  - Circuits with multiple clock domains
  - Internally derived clocks, mixed data and clock signals
  - Asynchronous logic
  - Circuits with combinational feedback paths
  - Embedded counters
  - Embedded RAMs and ROMs

Finite State Machine

- Example: serial adder
  - \( S_i = a_i \oplus b_i \oplus c_{i-1} \)
  - \( C_i = a_i b_i + c_{i-1} (a_i + b_i) \)
Example

- Corresponding parallel binary adder circuit
- Iterative network of the previous circuit

General Case

- Huffman model of sequential circuit
  - with edge-triggered D-flip-flops
General Case

- Any sequential circuit with edge-triggered D-FF can be directly converted into an iterative network

\[
x_0 \rightarrow y_0 \rightarrow Z_0
\]
\[
x_1 \rightarrow y_1 \rightarrow Z_1
\]
\[
x_r \rightarrow y_r \rightarrow Z_r
\]

Iterative Logic Array Expansion

- To detect a fault, a sequence of vectors may be needed
Example

Test for P SA0
- Provoke Fault on P1: a1 = 0, b1 = 1
- Propagate Fault to S2:
  - C0 = 1
    - Need to consider last time frame: a0 = 1, b0 = 1, Cin = X
  - a2 = 0, b2 = 0

Example

Test for u SA1
- In time frame t
  - provoke fault: u = 0
  - propagate fault effect to Z (path E, H, Z): v = 1 and G = 1
    - Justify B = C = 1: y = 1 and w = 1
  - Requires G = 1 in the time frame t-1
    - for a don’t care value of y (i.e., y = 0 or 1) in the time frame t - 1
Example (cont)

- Test for u SA1
  - In time frame $t - 1$
    - $G = 1$, $y = X$
    - $A = 1$
    - $U = D' \Rightarrow A = 0$ in the faulty circuit
  - Conflict!

- Problem with this example
  - Try to extend D-algorithm for sequential circuits
  - $Z$ is 1 in the presence of $u/1$ irrespective of
    - the logic values on other signal lines, and
    - the content of the flip-flop
Example (cont)

- Input sequence to set $Z$ to 0 in the fault-free circuit
  - $E = 0$ and $G = 1$
  - $B = 1 \Rightarrow y = 1$
    - $u = v = 0$ and $w = 1$ in cycle $t - 1$
    - $u = 0$, $v = 1$ and $w = 1$ in cycle $t$

- $u \quad CK \quad Z$
  - $v$
  - $w$

Nine-Valued Signals

- A fault can be detected even if in the presence of the fault a signal line in the faulty circuit has an unknown value (X)
  - While the corresponding signal line in the fault-free circuit has a known value (0 or 1) or vice-versa

- This information is not expressed by the logic values 0, 1, D and $D'$ introduced in the context of the D-algorithm
Nine-Valued Signals

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Symbol</th>
<th>Fault-free circuit value</th>
<th>Faulty Circuit Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0, 0&gt;</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;1, 1&gt;</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&lt;1, 0&gt;</td>
<td>D</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>&lt;0, 1&gt;</td>
<td>D'</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>&lt;X, X&gt;</td>
<td>X</td>
<td>0 or 1</td>
<td>0 or 1</td>
</tr>
<tr>
<td>&lt;0, X&gt;</td>
<td>G0</td>
<td>0</td>
<td>0 or 1</td>
</tr>
<tr>
<td>&lt;1, X&gt;</td>
<td>G1</td>
<td>1</td>
<td>0 or 1</td>
</tr>
<tr>
<td>&lt;X, 0&gt;</td>
<td>F0</td>
<td>0 or 1</td>
<td>0</td>
</tr>
<tr>
<td>&lt;X, 1&gt;</td>
<td>F1</td>
<td>0 or 1</td>
<td>1</td>
</tr>
</tbody>
</table>

Using Nine-valued Signals

- Propagate assigned values
- Assign values to propagate D or D
- Assign values to provoke D or D at stuck fault gate output
  - Primitive D-cube of the fault
- Line Justification
Propagating Assigned Values

<table>
<thead>
<tr>
<th>NOT</th>
<th>AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>D'</td>
</tr>
<tr>
<td>D'</td>
<td>D</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>G0</td>
<td>G1</td>
</tr>
<tr>
<td>G1</td>
<td>G0</td>
</tr>
<tr>
<td>F0</td>
<td>F1</td>
</tr>
<tr>
<td>F1</td>
<td>F0</td>
</tr>
</tbody>
</table>

**Propagating assigned values for AND and OR gates:**

- For propagating the fault effect through an OR gate with D input, apply \(<X, 0>\) to the other inputs of the OR gate.
- For propagating the fault effect through an OR gate with D’ input, apply \(<0, X>\) to the other inputs of the OR gate.
- For propagating the fault effect through an AND gate with D input, apply \(<1, X>\) to the other inputs of the AND gate.
- For propagating the fault effect through an AND gate with D’ input, apply \(<X, 1>\) to the other inputs of the AND gate.

Propagation D-Cubes

- For propagating the fault effect through an OR gate with D input, apply \(<X, 0>\) to the other inputs of the OR gate.
- For propagating the fault effect through an OR gate with D’ input, apply \(<0, X>\) to the other inputs of the OR gate.
- For propagating the fault effect through an AND gate with D input, apply \(<1, X>\) to the other inputs of the AND gate.
- For propagating the fault effect through an AND gate with D’ input, apply \(<X, 1>\) to the other inputs of the AND gate.
Assigning Values To Provoke $D$ Or $\overline{D}$

- At stuck fault gate output
- Primitive $D$-cube of the fault

<table>
<thead>
<tr>
<th>AND Gate with output $SA_0$</th>
<th>$a$</th>
<th>$b$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$&lt;1,X&gt;$</td>
<td>$&lt;1,X&gt;$</td>
<td>$D$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND Gate with output $SA_1$</th>
<th>$a$</th>
<th>$b$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$&lt;0,X&gt;$</td>
<td>$&lt;X,X&gt;$</td>
<td>$\overline{D}$</td>
</tr>
<tr>
<td></td>
<td>$&lt;X,X&gt;$</td>
<td>$&lt;0,X&gt;$</td>
<td>$\overline{D}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND Gate with input $a$ $SA_0$</th>
<th>$a$</th>
<th>$b$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$&lt;1,X&gt;$</td>
<td>$&lt;1,X&gt;$</td>
<td>$D$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND Gate with input $a$ $SA_1$</th>
<th>$a$</th>
<th>$b$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$&lt;0,X&gt;$</td>
<td>$&lt;X,1&gt;$</td>
<td>$\overline{D}$</td>
</tr>
</tbody>
</table>

Line Justification

<table>
<thead>
<tr>
<th>AND Gate with output $&lt;X,1&gt;$</th>
<th>$a$</th>
<th>$b$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$&lt;X,1&gt;$</td>
<td>$&lt;X,1&gt;$</td>
<td>$&lt;X,1&gt;$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND Gate with output $&lt;0,X&gt;$</th>
<th>$a$</th>
<th>$b$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$&lt;X,X&gt;$</td>
<td>$&lt;0,X&gt;$</td>
<td>$&lt;0,X&gt;$</td>
</tr>
<tr>
<td></td>
<td>$&lt;0,X&gt;$</td>
<td>$&lt;X,X&gt;$</td>
<td>$&lt;0,X&gt;$</td>
</tr>
</tbody>
</table>
Example: \( u/1 \)

- Provoke the fault
  - Apply \(<0, 1> = D'\) on signal line \( u \) at time frame \( t \)
- Propagate fault effect (along path through \( E \) and \( H \) to \( Z \))
  - \( v \) and \( G \) must be \(<1, X> = G_1\)
  - \( v \) is a primary input \( \Rightarrow v = <1, 1> \) at time \( t \)
    - \( A = <0, 0>\)
    - justify \( B = C = <1, X> \) (= \( G_1 \))
      - justifying \( C = <1, X> \) \( \Rightarrow w = C_1 <1, 1> \) at time frame \( t \)
      - justifying \( B = <1, X> \) \( \Rightarrow \) justify \( y = <1, X>\)
        - justify \( G = <1, X> \) at time frame \( t-1 \)

![Diagram of Digital System](image)

- Test for \( u/1 \)
  - \( u = 0, v = 0 \) and \( w = 1 \) in time frame \( t-1 \)
  - \( u = 0, v = 1 \) and \( w = 1 \) in time frame \( t \)
**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 $d_{seq} + 1$ time-frames
      - $d_{seq}$ is the sequential depth.
    - Cyclic circuit – Contains feedback among flip-flops:
      - May need $9^{Nff}$ time-frames
      - $Nff$ is the number of flip-flops.
  - Asynchronous circuit – Higher complexity!

- **Cycle-Free Circuits**
  - Characterized by
    - Absence of cycles among flip-flops and
    - a sequential depth, $d_{seq}$.
  - $d_{seq}$ 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 $d_{seq} + 1$. 

$max = \text{Number of distinct vectors with 9-valued elements} = 9^{Nff}$
**Cycle-Free Example**

Circuit:

- F1
- F2
- F3

- Level = 1
- Level = 1
- Level = 1

- dseq = 3

**s-graph**

- F1
- F2
- F3

- Level = 1
- Level = 1
- Level = 1

**All faults are testable**

---

**Cyclic Circuit Example**

Modulo-3 counter:

- CNT
- F1
- F2

**s-graph**

- F1
- F2

---
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 $9^{N_{ff}} = 81$, 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.

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 $d_{seq}$ time-frames
    - $d_{seq}$ is the maximum number of flip-flops on any path between PI and PO
  - Always initializable
- Cyclic circuits:
  - May need $9^{N_{ff}}$ time-frames
    - $N_{ff}$ is the number of flip-flops
  - Circuit must be initializable
  - Partial scan can make circuit cycle-free
- Asynchronous circuits:
  - High complexity
  - Low coverage and unreliable tests