Iterative Logic Arrays (ILAs)

- Some circuits are easy to test, for example, ILAs
- **Definition**: An ILA is a $k$-dimensional array-like circuit composed of identical cells with uniform interconnections
- Array circuits can be tested for powerful fault models using relatively few tests
- Examples
  - Arithmetic circuits
  - Ripple-carry adders
  - Array multipliers
  - Bit-sliced processors
  - Random-access memories: RAMs, ROMs
  - ILA models of sequential circuits

Example 1: Ripple-Carry Adder

- 1D array composed of full-adder cells
Example 1: Ripple-Carry Adder

- Assume the cell fault (CF) model, which implies that all SLL faults in all realizations will be detected
- We must apply eight patterns to every cell and observe the responses
- Six of the 8 patterns can be applied simultaneously to all cells, e.g. $A_iB_iC_i = 000$

    \[
    \begin{array}{cccccc}
    & 0 & 0 & 0 & 0 & 0 \\
    \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
    0 & 0 & 0 & 0 & 0 \\
    \end{array}
    \]

- Faults in $FA_i$ can be observed via $S_i$ or $S_{i+1}$
- Two of the 8 patterns cannot be applied simultaneously to all cells, namely $A_iB_iC_i = 001$ and $110$, because $C_{in} \neq C_{out}$

Example 1: Ripple-Carry Adder

- The patterns $A_iB_iC_i = 001$ and $110$ can be applied simultaneously to alternating cells

    \[
    \begin{array}{cccccc}
    1 & 1 & 0 & 0 & 1 & 1 \\
    \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
    1 & 0 & 1 & 0 & 1 \\
    \end{array}
    \]

- All CF faults in an $n$-bit RC adder can be detected by 8 tests, independent of the array size $n$
- The property of an $n$-cell ILA that all (cell) faults be detected by a constant number test patterns for any $n$ is called C-testability
Example 2: Gate Array

- This is an ILA realization of a $k$-input AND function, $k = 1, 2, 3, \ldots$
- **Question 1:** Is an AND array C-testable?

- **Question 2:** What if the AND function is replaced by XOR?

Design for Testability (DFT)

- Test generation algorithms for logic circuits are complex (NP complete)
- Circuits containing, say, $10^6$ gates or $10^2$ flip-flops, may be too large for current ATPG tools
- Heuristic methods are used for testing complex circuits as we saw for microprocessors, RAMs, etc.
- Fault coverage of such methods can be low and hard to determine
- To ensure high levels of testability, design for testability (DFT) is often essential
**Testability Goals**

- Maximize fault coverage:
  \[ \left( \frac{\text{no. of faults detected}}{\text{no. of possible faults}} \right) \times 100\% \]
- Minimize test application time
- Minimize test data size
- Minimize test generation effort

**Testability Measures**

- Insufficient controllability or observability is the key reason why most circuits are hard to test.
- This suggests computing controllability and observability values for all signal nodes in a circuit. The “bad” values can pinpoint hard-to-test areas of the circuit that need redesign.
- **Basic idea:** estimate how hard it is to apply or observe a 0/1 on each node of the circuit.
- The computation effort should be small, for example, 10% of the corresponding ATPG effort. This implies the use of heuristic measures.
- Many related testability measuring schemes have been proposed,
  - TMEAS [Stephenson and Grason, 1976]
  - SCOAP [Goldstein 1979]
  - COP [Brglez, 1984]
**SCOAP**

- Associates two controllability numbers \( C^0, C^1 \) and one observability (O) number with each circuit node; larger values are worse
- Treats a node as sequential (S) if it is a memory element output, otherwise it is combinational (C)
- Primary inputs: \( CC^{0/1} = 1 \), \( SC^{0/1} = 0 \); primary outputs: \( CO = SO = 0 \)
- Example: NAND gate with inputs \( x_1, x_2, \ldots, x_n \), and output \( z \)

  Calculate controllability nos. by going from inputs to outputs
  
  \[
  CC^0(z) = \sum_i CC^1(x_i) + 1 \\
  SC^0(z) = \sum_i SC^1(x_i) \\
  CC^1(z) = \min \{CC^0(x_i)\} + 1 \\
  SC^1(z) = \min \{SC^0(x_i)\}
  \]

  Calculate observability nos. by going from outputs to inputs
  
  \[
  CO(x_i) = CO(z) + \sum_{i \neq j} CC^1(x_j) + 1 \\
  SO(x_i) = SO(z) + \sum_{i \neq j} CCS(x_j)
  \]

**Example:** 7-stage feedback shift register
Example (contd): Controllability profiles

Example (cont’d): Observability profiles
**Testability vs. Testing Method**

Consider testing a large combinational circuit. The impact of the testing method depends on the circuit size, complexity, and practical design constraints.

<table>
<thead>
<tr>
<th>Testability parameter</th>
<th>Testing method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PODEM</td>
</tr>
<tr>
<td>Fault coverage</td>
<td>good</td>
</tr>
<tr>
<td>Test data length</td>
<td>best</td>
</tr>
<tr>
<td>Test generation effort</td>
<td>worst</td>
</tr>
<tr>
<td>Suitability for BIST</td>
<td>bad</td>
</tr>
</tbody>
</table>

**DFT Approaches**

**Ad Hoc Design Rules**
- Insert control/observation points (“test points”), e.g. a reset line
- Avoid redundancy
- Improve circuit structure, e.g. break global feedback during testing
- Provide clock access during testing

**Systematic Design Methods**
- Specialized logic design techniques
- Regularization
- Scan design
- Various BIST techniques
Test Point Insertion

- Make hard-to-control internal signals controllable via extra primary inputs and logic (control points)
- Make hard-to-observe internal signals observable via extra primary outputs and logic (observation points)

\[ y^* = y \text{ when } x = a \]

Test Point Insertion

Control-Point Sites

- Clock lines
- Global reset lines
- Inputs of state-control devices
- Lines of high fanout (fanout stems)
- Control (especially tristate control) lines of buses
- All bus lines in bus-structured designs
- Control inputs to (embedded) RAMS and ROMs
- Some enable/hold/select control lines
- Lines identified by testability measuring programs as having low controllability
**Test Point Insertion**

**Observation-Point Sites**
- “Buried” (not directly accessible) control/status lines
- Outputs of state-control devices
- Lines of high fanout (fanout stems)
- Outputs of “funneling” circuits (high fanin), e.g. parity generators
- Logically redundant nodes
- Global feedback paths
- Output lines in bus-structured designs
- Lines identified by testability measuring programs as having low observability

**Main Limitation**
- Availability of (spare) input/output terminals

---

**Test Point Insertion**

**Memory Control/Observation**

![Diagram showing memory control/observation with flip-flops and reset/shift conditions]
**DFT: Circuit Restructuring**

**Counter Design**

- n-bit counter
- Stage 1 k-bit counter
- Stage 2 k-bit counter
- Stage m k-bit counter
- Count
- Reset
- m control points
- x1, xm

**Controlling Feedback**

- Combinational circuit C
- Microcontroller
- Loopback
**DFT: Timing Control**

- Avoid asynchronous timing
- Make clocks observable and controllable during testing

---

**Design Rule Summary**

- Partition large hard-to-test circuits into small testable components
- Design controllable (e.g. initializable) and observable units with careful selection of control/test points
- Allow global feedback paths to be opened/closed
- Avoid redundancy, or allow it to be overridden during testing
- Avoid asynchronous circuits and provide access to clock signals

**Conclusions**

- Often ad hoc design modification is too late to significantly improve a circuit's testability
- Develop a systematic test plan at the start of the design process