Design For Testability - Organization

- Overview of DFT Techniques
- Ad-hoc techniques
  - Examples
  - I/O Pins
- Scan Techniques
  - Full & Partial Scan
  - Multiple Scan Chains
- Boundary Scan
- Built-In Self-Test
- Evaluation Criteria for DFT Techniques
Overview of DFT Techniques

- **Ad-hoc techniques**
  - Target “difficult-to-test” subcircuits to improve
    - Controllability
    - Observability

- **Scan techniques**
  - Up-front & top-down structured techniques
  - Enforce general design style & require following design rules
  - CAD tools for automatic implementation & vector generation
  - Full scan design - best overall DFT technique
  - Partial scan design
  - **Boundary scan** - for testing device I/O interconnect

- **Built-In Self-Test**
  - Internal test circuitry incorporated within chip/board
Ad-Hoc DFT Techniques: Basic Idea

- Add MUXs to provide access to/from internal circuitry
  - Controllability & Observability
- Add gates to provide control to internal circuitry
  - Controllability only
- Add these “test points” only where needed in circuit
  - Low area overhead penalty
  - Little (if any) performance impact
    - Critical paths can often be avoided
  - Target difficult to test subcircuits
    - Potential for significant increase in fault coverage
- Creative testability solutions on a case-by-case basis
  - But we have to figure out what & where those are
Ad-Hoc DFT Techniques: Some Benefits

- Provide test points for controllability & observability
- Provide easier initialization
  - For logic simulation and design verification
- Partition the logic into easier to test pieces
  - Provide access to embedded blocks
  - Core tests can be re-used
- Bypass clock generation ckts (oscillators, one-shots, etc.)
- Avoid or bypass asynchronous logic
- Break feedback loops (when they are a problem)
- Break up large counters into smaller ones
- Disable intentional redundant logic for testing
Ad-Hoc Techniques: Examples

- Add pin(s), MUX(s), gate(s) to change mode of operation
- Use test mode to modify structures detrimental to test
  - Breaking un-initializable feedback loops
    - Helps prevent potentially detectable faults
  - Bypassing internal oscillators
Ad-Hoc Techniques: Examples (cont.)

- Sometimes gates can replace MUX (gate is smaller)
  - Asynchronous resets/presets
    - MUX to bypass vs. gate to block
  - Breaking up large counters
    - MUX/gate in carry-chain
    - Reduces test time from $2^N$ to $2^{N/2}$

\[\begin{array}{c}
\text{Cin} \quad \text{Counter} \quad \text{Cout} \\
0 \quad 1
\end{array}\]
Ad-Hoc Techniques: Examples (cont.)

Partitioning into easy to test subcircuits using MUXs

- Each subcircuit can be tested independently
  - this may require many MUXs

- Example of partitioning pipelined structure (works for almost any ckt)
  - T1=0, T2=1 ⇒ normal system mode of operation
  - T1=0, T2=0 ⇒ testing ckt A
  - T1=1, T2=0 ⇒ testing ckt B
  - T1=1, T2=1 ⇒ testing ckt C
Ad-Hoc Techniques: Examples (cont.)

Partitioning using fewer MUXs

- Testing is more difficult but easier than original circuit
  - Less than full DFT
- A more observable
  - Less than full DFT
- C more controllable
  - Less than full DFT
- B is completely
  - Controllable
  - Observable
  - Same as full DFT
- Half the area penalty
Ad Hoc Techniques: I/O Pins

Extra I/O pins are often required (sometimes a lot of I/O)

- Test data inputs may be able to share primary inputs
- Test data outputs can share primary outputs
- Test data/mode for gate test point typically need I/O pins
- Test mode control signals for MUX test points require:
  - Extra I/O pins
  - Devices with out processor interface
  - Decode test mode pins to obtain desired test modes
  - Assumes only a subset of possible combinations needed

- Extra internal test register bits
  - Devices with processor interface

C. Stroud 9/09
Ad-Hoc Techniques: Summary

**Advantages:**
- Relatively low area overhead and performance impact
  - 1 MUX (or gate) per test point
  - Critical paths can often be avoided
- Moderate to good improvements in testability
- Does not constrain the design
- Can be used with other DFT techniques like BIST

**Disadvantages:**
- Requires manual design & implementation
  - Limited CAD support (if any) available
- I/O pin overhead can be high in some cases
- Fault simulation required to evaluate effectiveness
- Considerable test development effort required
Scan Design Techniques: Basic Idea

- Transform FFs in sequential logic into shift register
  - All FFs controllable & observable via serial access
    - Many different types of scan FF (too many!)
  - Test problem simplified to testing combinational logic
    - The best overall DFT approach ever developed
Level Sensitive Scan Design

- Originally developed by IBM
  - Used level sensitive latches
  - System vs. scan mode controlled by clocks
    - SCLK – system clock to slave latch
    - MCLK – system clock to master latch
      - 1 MCLK pulse + 1 SCLK pulse = 1 system clock cycle
    - TCLK – test clock to scan latch
      - 1 TCLK pulse + 1 SCLK pulse = 1 scan clock cycle

![Diagram of Level Sensitive Scan Design](image)
Scan Design Testing Sequence

Step 1. Shift some patterns through scan chain
   - FFs are now completely tested
Step 2. Scan-in test vector
   - Scan Mode = 1
Step 3. Apply test vector at PIs
Step 4. Observe results at POs
Step 5. Apply Clock to capture results in scan chain
   - Scan Mode = 0
Step 6. Scan-out results & scan-in next test vector
   - Scan Mode = 1
Step 7. Go to Step 3 until all test vectors processed
Scan Design Techniques: Advantages

- Fully automated DFT process
  - Well-supported by CAD vendors
  - Sequential circuits become combinational in test mode
    - ATPG CAD tools for generating test vectors
      - Typically no fault simulation required
    - ATPG can identify redundant logic
      - Helps minimize design
  - Reduced time-to-market
- High fault coverage
  - Near 100% for gate level stuck-at and bridging faults
- Can be applied hierarchically
  - chips ⇒ boards ⇒ system
- Allows simplified & accurate fault/defect diagnosis & FMA
- Highly structured & provides good basis for BIST
Scan Design Techniques: Disadvantages

- **Overhead:**
  - **Pins:** 3 (Scan Data In, Scan Data Out, Scan Mode)
    - *Scan Data In* can be shared with primary input
    - *Scan Data Out* can be shared with primary output
  - **Area Overhead:** 1 MUX per flip-flop
    - Typically 2-10% total area overhead penalty
  - **Performance degradation:** added delay through MUX
  - **Routing overhead:** scan chain connections, *Scan Mode*

- **Long test application time:**
  - 
    - # clock cycles = #vectors × (#FFs + 1) + #FFs
      - Lots of vectors & output responses to store
  - **Logic must be synchronous**
  - **Difficult to test at system clock speed**
Variations on Scan Design Techniques

- Multiple scan chains
  - Fewer clock cycles to scan vectors in/out
  - Reduces test time if scan chains are balanced
    - $M$ times faster for $M$ equal length scan chains
  - Good for multiple clock (or clock-edge) circuits
  - Additional pins for Scan Data In/Out for each chain
    - Scan Data In can come from PIs
    - Scan Data Out can use POs if output FF is last in scan chain

- Partial Scan Design replaces only selected FFs in device
  - Full scan replaces all FFs in device with scan FFs
  - Lower area & performance penalty than full scan
    - But usually lower fault coverage as well
Scan Design Techniques: Partial Scan

Lower area & performance penalty comes with price:

- What FFs should be replace and how many
  - Trade-off between fault coverage and area overhead
  - FF replacement selection methods
    - **Structural**: select FFs to cut loops
    - **ATPG-based**: select FFs useful during ATPG
    - **Testability-based**: select FFs to maximize testability

- CAD tool support more limited than for full scan
- Fault simulation typically required for partial scan
  - Not required for full scan since done by ATPG
- Significantly more complex diagnosis
- Still difficult test at system clock speed
Full Scan - Partial Scan Comparison Example

Comparison study data from *Jet Propulsion Laboratory*

- Viterbi Butterfly Decoder used for comparison
  - Fewer FFs replaced with partial scan
  - Lower area overhead with partial scan
  - More vectors but fewer clock cycles with partial scan

<table>
<thead>
<tr>
<th></th>
<th>Full Scan</th>
<th>Partial Scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Scan FFs</td>
<td>448</td>
<td>256</td>
</tr>
<tr>
<td>Logic Overhead</td>
<td>24.6%</td>
<td>14.1%</td>
</tr>
<tr>
<td>Number of Vectors</td>
<td>34</td>
<td>41</td>
</tr>
<tr>
<td>Clock cycles/vector</td>
<td>449</td>
<td>262</td>
</tr>
<tr>
<td>Total clock cycles</td>
<td>15,714</td>
<td>10,998</td>
</tr>
</tbody>
</table>
Boundary Scan

● Developed to test interconnect between chips on PCB
  ◆ Originally referred to as JTAG (Joint Test Action Group)
  ◆ Uses scan design based approach to test external interconnect
  ◆ No-contact probe overcomes problems of “in-circuit” testing:
    ➢ Surface mount components with less than 100 mil pin spacing
    ➢ Double-sided component mounting
    ➢ Micro- and floating vias

● Provides standardized test interface
  ◆ IEEE standard 1149.1
  ◆ Four wire interface
    ➢ TMS - Test Mode Select
    ➢ TCK - Test Clock
    ➢ TDI - Test Data In
    ➢ TDO - Test Data Out
    ➢ TRST - reset (optional & rarely included)
Boundary Scan (cont.)

Additional logic required:
- 1 Boundary Scan cell per I/O pin
- Test Access Port (TAP)
  - 4-wire interface
    - TMS
    - TCK
    - TDI
    - TDO
  - TAP controller
    - 16-state FSM
    - Controlled by TMS & TCK
  - various registers for
    - Instructions
    - Operations
Boundary Scan Cell Architecture

BS Cell Operation

<table>
<thead>
<tr>
<th>Operational Mode</th>
<th>Data Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Scan</td>
<td>IN → OUT</td>
</tr>
<tr>
<td>Capture</td>
<td>$S_{IN}$ → CAP</td>
</tr>
<tr>
<td>Update</td>
<td>IN → CAP</td>
</tr>
<tr>
<td></td>
<td>CAP → UPD</td>
</tr>
</tbody>
</table>

Bi-directional buffers require multiple BS cells

BS test data in ($S_{IN}$)

Input data to IC core

Tri-state control From IC core

Output data From IC core

BS test data out ($S_{OUT}$)
Boundary Scan TAP Controller Operation

1. Send test instruction serially via 
   \textit{TDI} into Instruction Register \textit{(shift-IR)}

2. Decode instruction and configure test circuitry \textit{(update-IR)}

3. Send test data serially into Data Register \textit{(shift-DR)} via \textit{TDI}

4. Execute instruction \textit{(update-DR \& capture-DR)}

5. Retrieve test results captured in Data Register \textit{(shift-DR)} serially 
   via \textit{TDO}

\textit{Note: transitions on rising edge of TCK based on TMS value}

C. Stroud 9/09 Design for Testability
Boundary Scan Instructions

Defined by IEEE 1149.1 standard:

- **Mandatory Instructions**
  - Extest – to test external interconnect between ICs
  - Bypass – to bypass BS chain in IC
  - Sample/Preload – BS chain samples external I/O
  - IDCode – 32-bit device ID

- **Optional Instructions**
  - Intest – to test internal logic within the IC
  - RunBIST – to execute internal Built-In Self-Test
    - If applicable (this is rare)
  - UserCode – 32-bit programming data code
    - For programmable logic circuits
  - User Defined Instructions
Boundary Scan: User-Defined Instructions

- User-defined instructions facilitate:
  - Public instructions (available for customer use)
  - Private instructions (for the manufacturer use only)
  - Extending the standard to a universal interface
    - For any system operation feature or function
    - A communication protocol to access new IC test functions

- Example: *Texas Instruments 74BCT8244*
  - Octal buffer with Boundary Scan
  - Additional 2-bit Control Register whose state can reconfigure the BS Register for BIST functions (pseudorandom pattern generator or signature analyzer)
  - Additional instructions to initiate these functions
Boundary Scan: Advantages

- It’s a standard!!! (IEEE 1149.1)
  - Allows mixing components from different vendors
  - Provides excellent interface to internal BIST circuitry
- Well supported by CAD tool vendors, IC & ATE manufacturers
- Allows testing of board & system interconnect
  - Back-plane interconnect test without using PCB functionality
  - Very high fault coverage for interconnect
- Useful in diagnosis & FMA
  - Provides component-level fault isolation
  - Allows real-time sampling of devices on board
  - Useful at wafer test (fewer probes needed)
- BS path reconfigured to bypass ICs not under test for faster test
Boundary Scan: Disadvantages

- **Overhead:**
  - Logic: about 300 gates/chip for TAP + about 15 gates/pin
    - Overall overhead typically small (1-3%)
    - But significant for only testing external interconnect
      - Especially tri-state (2 cells) & bi-directional buffers (3 cells)
  - I/O Pins: 4
    - 5 if optional TRST (Test Reset) pin is included
  - I/O delay penalty
    - 1 MUX delay on all input & output pins
      - This can be reduced by design

- Internal scan design cannot have multiple chains
- Cannot test at system clock speed
  - But internal BIST can run at system clock speed
Built-In Self-Test (BIST)

- Provides the capability of a circuit to test itself
- Can be applied hierarchically: module, chip, board, or system
- Provides *vertical testability* = same test circuitry used all levels of testing: from chip to system
- **On-line BIST:** testing occurs during normal system operation
- **Off-line BIST:** testing occurs when circuit is out-of-service
### DFT Evaluation Criteria

- Area overhead
- Performance penalties
- IO pin count
- CAD tool support
- Fault simulation
- ATE cost
- Power dissipation
- Risk to project

<table>
<thead>
<tr>
<th>Area Overhead Calculation Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>chip area</td>
<td>area of chip w/DFT</td>
</tr>
<tr>
<td></td>
<td>area of chip w/o DFT</td>
</tr>
<tr>
<td>number of gates</td>
<td>total gates w/DFT</td>
</tr>
<tr>
<td></td>
<td>total gates w/o DFT</td>
</tr>
<tr>
<td>frequently used method</td>
<td>total gates for DFT</td>
</tr>
<tr>
<td></td>
<td>total gates w/o DFT</td>
</tr>
<tr>
<td>most frequently used method</td>
<td>total gates for DFT</td>
</tr>
<tr>
<td></td>
<td>total gates w/DFT</td>
</tr>
<tr>
<td>number of gate inputs</td>
<td>total gate inputs for DFT</td>
</tr>
<tr>
<td></td>
<td>total gate inputs w/DFT</td>
</tr>
<tr>
<td>Number of gate I/O</td>
<td>total gate I/O for DFT</td>
</tr>
<tr>
<td></td>
<td>total gate I/O w/DFT</td>
</tr>
</tbody>
</table>

- Increase in design time vs. test time reduction
- Economic impact on product
  - Impact on product quality and product cost
- How well does the DFT circuitry get tested?
  - Does the BIST circuitry also test itself?