Design-For-Testability

Alberto Bosio

bosio@lirmm.fr

How to improve the fault coverage?
How to improve the fault coverage?

Uncollapsed Stuck Fault Summary Report

<table>
<thead>
<tr>
<th>fault class</th>
<th>code</th>
<th>#faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detected</td>
<td>DT</td>
<td>10</td>
</tr>
<tr>
<td>Undetectable</td>
<td>UD</td>
<td>28</td>
</tr>
<tr>
<td>total faults</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>fault coverage</td>
<td></td>
<td>35.71%</td>
</tr>
<tr>
<td>test coverage</td>
<td></td>
<td>100.00%</td>
</tr>
</tbody>
</table>

How to improve the fault coverage?

- The 28 Undetectable faults are due to the circuit structure
  - The circuit is hard to test

**Idea**

- Modify the circuit structure in order to make it more easy to test
- The functionality is exactly the same
- We perform a so called Design-for-Testability (DfT)
How to improve the fault coverage?

We can add one more primary output to make faults observable.

How to improve the fault coverage?

Used only during the test.
How to improve the fault coverage?

Uncollapsed Stuck Fault Summary Report

<table>
<thead>
<tr>
<th>fault class</th>
<th>code</th>
<th>#faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detected</td>
<td>DT</td>
<td>34</td>
</tr>
<tr>
<td>Undetectable</td>
<td>UD</td>
<td>6</td>
</tr>
<tr>
<td>total faults</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>fault coverage</td>
<td></td>
<td>85.00%</td>
</tr>
<tr>
<td>test coverage</td>
<td></td>
<td>100.00%</td>
</tr>
</tbody>
</table>

How to improve the fault coverage?

- Ad-hoc technique
  - it requires experience
  - not automated
  - it may have high cost
    - extra pins, time to identify the problem, ...
  - it works!
Sequential Circuits

Primary Output network

Next-state network

State register

clk
reset

Primary Output network

Next-state network

State register

clk, reset

N FlipFlops
Sequential Circuits

- A sequential circuit has memory in addition to combinational logic.
- Test for a fault in a sequential circuit is a sequence of vectors, which
  - Initializes the circuit to a **known state**
  - Activates the fault, and
  - Propagates the fault effect to a primary output
- Methods of sequential circuit ATPG
  - Time-frame expansion methods
  - Simulation-based methods

Un exemple « simple »

- Test du collage à zéro de la sortie de P
Concept of Time-Frames

- If the test sequence for a single stuck-at fault contains \( n \) vectors,
  - Replicate combinational logic block \( n \) times
  - Place fault in each block
  - Generate a test for the multiple stuck-at fault using combinational ATPG

ATPG for sequential circuits

- Complex compared to ATPG for combinational circuits
- Low fault coverage (~30%)
- High number of test patterns

Sequential circuits are not easy to test!!
DfT for Sequential Circuits

State register

clk
reset

Primary Output network

Next-state network

SI
SO

Primary Output network

Next-state network

State Shift Register

SI
SO

clk
reset

TE
DfT for Sequential Circuits

- The DfT is called FULL SCAN
  - All the Flip Flops are now controllable and observable
    - I can shift in (scan in) any values I want
    - I can shift out (scan out) all the FlipFlops values
  - The circuit is now like a combinational one for the ATPG
DfT for Sequential Circuits

Scan Flip Flops

Master latch
Slave latch

D flip-flop

Logic overhead

MUX

Normal mode, D selected
Scan mode, SD selected

05/04/16
Routing Overhead

- IO pad
- Flip-flop cell
- Routing channels
- Interconnects

SCAN Benefits

- ATPG applied on s5378

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Full-scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of combinational gates</td>
<td>2,781</td>
<td>2,781</td>
</tr>
<tr>
<td>Number of non-scan flip-flops (10 gates each)</td>
<td>179</td>
<td>0</td>
</tr>
<tr>
<td>Number of scan flip-flops (14 gates each)</td>
<td>0</td>
<td>179</td>
</tr>
<tr>
<td>Gate overhead</td>
<td>0.0%</td>
<td>15.66%</td>
</tr>
<tr>
<td>Number of faults</td>
<td>4,603</td>
<td>4,603</td>
</tr>
<tr>
<td>PI/PO for ATPG</td>
<td>35/49</td>
<td>214/228</td>
</tr>
<tr>
<td>Fault coverage</td>
<td>70.0%</td>
<td>99.1%</td>
</tr>
<tr>
<td>Test Coverage</td>
<td>70.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>CPU time on SUN Ultra II, 200MHz processor</td>
<td>5,533 s</td>
<td>5 s</td>
</tr>
<tr>
<td>Number of ATPG vectors</td>
<td>414</td>
<td>585</td>
</tr>
</tbody>
</table>
How to apply the test?

Scan Chain Test

- Scan register must be tested prior to application of scan test sequences.
- A shift sequence 00110011 . . . of length $n_{\text{ff}}$ in scan mode (TE=0) produces 00, 01, 11 and 10 transitions in all flip-flops and observes the result at SCANOUT output.
Logic Test

- Vectors to test the combinational logic

Test time:
- \((nb\text{-test}) \times (nb\text{-FF})\)
Full Scan

- Overheads
  - Are, routing, test time
- + High fault coverage
- + Easy for the ATPG

Multiple Scan Chains
Un exemple « simple »

Solution triviale en utilisant une solution "scan total"