Lab2
Scan Chain Insertion and ATPG Using DFTADVISOR and FASTSCAN

Pro: Chia-Tso Chao
TA: Szu-Pang Mu
Chien Hsueh Lin
2015/05/26
Outline

- Introduction
- Dftadvisor
- Fastscan
- Mix Flow
- Lab
Outline

- Introduction
- DFTADVISOR
- FASTSCAN
- Mix Flow
- Lab
Introduction

- This lab focuses on ATPG results from different tools: Mentor Graphic and Synopsys.
- Dftadvisor is used to insert scan chain (basically replace FF with scan FF).
- Fastscan is used to do ATPG and fault simulation.
Insert Scan and ATPG Flow

Tool Flow

- **RTL Coding**
  - RTL Design

- **SYNTHESIS**
  - DRC
  - Scan Insertion
    - DFTAdvisor
  - Gate Level Netlist
    - Scan Inserted Netlist

- **ATPG**
  - FastScan
  - DRC

- **Test Patterns**
  - ATE
Input/Output Files

- **Simulation Library**: adk.atpg
- **Dftadvisor**: design_scan.v
- **Scan Chain Information**: design_scan.dofile, design_scan.testproc
- **Fastscan**
- **Circuit Netlist**: design.v
- **Simulation Testbenches**
- **ATE Vectors**
- **Fault Reports**
Outline

- Introduction
- DFTADVISOR
- FASTSCAN
- Mix Flow
- Lab
Invoke DFTADVISOR

- $ source /usr/cad/mentor/CIC/tessent.cshrc
  - Just enter above command for the first time invoke dftadvisor.

- Read in verilog source file and assign library file.
  - $ cd lab2
  - $ dftadvisor pre_norm_noscan.v -verilog -lib l90sprvt.atpg -nogui
Specify Clock

Clocks are primary input signals that asynchronously change the state of sequential logic elements.

- SETUP> add clock 0 clk

  specify which state cannot affect output
Setup Test Logic Configuration

- Set scan methodology.
  - Mux_scan : mux-DFF
  - Lssd : level sensitive DFF
  - Clocked_scan : clocked-signal
  - SETUP> set scan type m

- Test logic options-- make clock lines controllable to get a scannable design.
  - SETUP> set test logic -clock on -reset on
DRC

- Entering DRC mode.
  - SETUP> set system mode dft

- Setup scan type full scan.
  - DFT> setup scan identification full_scan

- Perform DRC.
  - DFT> run

- Report some information.
  - DFT> report statistics
DRC Rules

- General Rules
- Procedure Rules
- Scan Chain Trace Rules
- Scan Cell Data Rules
- Clock Rules
- Ram Rules
- BIST Rules
- EDT Rules
- Timing Rules
Insert Scan Chain and View Report

☐ Set # of scan-chains to insert and do so.
  ■ DFT> insert test logic -number 10

☐ Report scan-chain information.
  ■ DFT> report scan chain
  ■ DFT> report test logic
Output Scanned Design for ATPG

- Write out files and exit DFT ADVISOR
  - DFT> write netlist pre_norm_scan.v -verilog -replace
  - DFT> write atpg setup pre_norm_scan -replace
  - DFT> exit

.dofile : setup information
.testproc : procedure file
Outline

- Introduction
- DFTADVISOR
- FASTSCAN
- Mix Flow
- Lab
Invoke Fastscan

☐ Invoke Fastscan:

- source /usr/cad/mentor/CIC/dft.cshrc

☐ specify scanned verilog file and library file.

- $ fastscan pre_norm_scan.v -verilog -lib l90sprvt.atpg -nogui
Read Setup Information

- Read setup information from Dftadvisor.
  - SETUP> dofile pre_norm_scan.dofile
- Entering atpg mode.
  - SETUP> set system mode atpg
- Setup fault type: stuck, iddq, toggle, transition.
  - ATPG> set fault type stuck
Generate Patterns

- Uses '-auto' option to allow Fastscan to analyze design and suggest the best settings possible to generate the most compact patterns with the highest coverage with the lowest time.

  ATPG> create patterns -auto
Generate Patterns

- Without ‘auto’ option, you can specify your own configurations using these commands:
  - set atpg limits -Cpu_seconds [integer] -Test_coverage [real] -Pattern_count [integer]
  - set atpg compression on -Abort_limit [integer]
  - identify redundant faults

ATPG> create patterns
View Report

- Report simulation result and faults.
  - ATPG> report statistics
  - ATPG> report faults -all
Simulation performed for gates = 27550  faults = 110115

system mode = ATPG  pattern source = internal patterns

#patterns test  #faults #faults  # eff.  # test  process  RE/AU/abort
simulated coverage in list detected patterns patterns CPU time

deterministic ATPG invoked with comb/seq abort limit = 300/100

---

0.10 sec  224/0/0  32  32  0.15 sec
54 91.30%  10501  9161  32  64  0.26 sec
54 95.67%  5187  4395  32  96  0.41 sec
54 98.43%  1887  2450  32  128  0.60 sec
54 99.22%  954  516  10  138  0.68 sec
54 99.69%  357  597  32  170  0.69 sec
54 99.98%  24  333  32  202  0.70 sec
54 100.00%  0  24  3  205  0.70 sec
### Statistics report

<table>
<thead>
<tr>
<th>Fault Classes</th>
<th>#faults (total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FU (full)</td>
<td>6078</td>
</tr>
<tr>
<td>UO (unobserved)</td>
<td>9 (0.15%)</td>
</tr>
<tr>
<td>DS (det_simulation)</td>
<td>5393 (88.73%)</td>
</tr>
<tr>
<td>DI (det_implication)</td>
<td>540 (8.88%)</td>
</tr>
<tr>
<td>UU (unused)</td>
<td>16 (0.26%)</td>
</tr>
<tr>
<td>RE (redundant)</td>
<td>118 (1.94%)</td>
</tr>
<tr>
<td>AU (atpg_untestable)</td>
<td>2 (0.03%)</td>
</tr>
</tbody>
</table>

### Coverage

<table>
<thead>
<tr>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_coverage</td>
</tr>
<tr>
<td>fault_coverage</td>
</tr>
<tr>
<td>atpg_effectiveness</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#test_patterns</th>
<th>#simulated_patterns</th>
<th>CPU_time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>224</td>
<td>3.7</td>
</tr>
</tbody>
</table>
Fault Types (1/3)

- AU: Atpg_untestable
  - Due to pin constraint or insufficient sequential depth cause testable faults become atpg_untestable

- FU: Full
- TE: Testable
- DT: Detected
Fault Types (2/3)

- **UT: Untestable**
  - Faults which no pattern can exist to either detect or possible-detect them, such as unused pins

- **UD: Undetected**
  - Faults cannot be proven untestable or ATPG_untestable

- **RE: Redundant**
Fault Types(3/3)

- **UU**: unused
  - All faults unconnected to any circuit observation point

- **BL**: blocked
  - Faults which logic blocks all paths to an observation point

- **TI**: tied
  - Point of the fault value is always same (and-gate with complementary inputs)
Test Coverage Formula Comparison

- **Tmax**

  \[
  \text{test\_coverage} = \frac{DT + (PT \times \text{posdet\_credit})}{\text{all\_faults} - (UD + AU \times \text{au\_credit})} \times 100
  \]

- **Fastscan**

  \[
  \text{test\_coverage} = \frac{DT + (PT \times \text{posdet\_credit})}{\text{testable}} \times 100
  \]

  \[
  \text{fault\_coverage} = \frac{DT + (PT \times \text{posdet\_credit})}{\text{full}} \times 100
  \]

  \[
  \text{ATPG\_effectiveness} = \frac{DT + UT + AU + PU + (PT \times \text{posdet\_credit})}{\text{full}} \times 100
  \]

- **Testable** = DT + PT + AU + UD

- **Untestable** = UU + TI + BL + RE
Save Patterns

- Save patterns just generated. The response of ATE saved in s38584_seq_ate.pat
- Various format including binwgl, ctl2005, stil2005, stil999, verilog, vhdl, wgl, zycad, tstl2, utic.
  - ATPG> save patterns pre_norm_scan.pat -verilog -proc -replace
  - ATPG> save patterns pre_norm_scan_tstl2.pat -TSTL2 -rep
  - ATPG> exit

Toshiba Standard Tester Interface Language 2
Testing Flow

- Synopsys Design Compiler is better in mapping from RTL code to gate-level circuit.
- Some cases in industry field uses Design Compiler to synthesis RTL code and uses Fastscan to perform ATPG and fault simulation.
Input/Output Files

Simulation Library

tsm18.v

DC

Scan Chain Information

design_scan.stil

Circuit Netlist

design.v

Scanned Circuit

design_scan.v

Fastscan

libcomp

stil2mgc

Simulation Testbenches

ATE Vectors

Fault Reports
Input Files Required from Design Compiler’s Flow

- Library file need to be converted.
  - From .v to .atpg
- The detail information about scanned circuit need to be converted.
  - From .stil to .dofile
- Scanned code.
Fastscan uses different library from Tetramax, so use 'libcomp' command to convert l90sprvt.v to l90sprvt.atpg.

- $ libcomp l90sprvt.v
- SETUP> add model -all
- SETUP> set optimization on
- SETUP> set learning on
- SETUP> set sys mode tran
- TRANSLATION> run
- TRANSLATION> write lib l90sprvt_test.atpg -rep
- TRANSLATION> exit
Convert STIL File

- Using command 'stil2mgc' to convert STIL file into dofile and test procedure file for Fastscan.
  - $ stil2mgc pre_norm_scan.stil
  - It will generate pre_norm_scan.stil.do and pre_norm_scan.stil.proc
Performing ATPG using FASTSCAN

- Read scanned circuit and library from design compiler to perform ATPG.
  - `$ fastscan pre_norm_scan.v -verilog -lib l90sprvt.atpg -nogui`
  - SETUP> dofile pre_norm_scan.stil.do
  - SETUP> set sys mode atpg
  - ATPG> create patterns -auto
  - ATPG> report statistics
Outline

- Introduction
- DFTADVISOR
- FASTSCAN
- Mix Flow
- Lab
Lab Goal

- Compare test coverage and # of patterns and run time during ATPG using methods in DC + TMAX and DFTA + FS and DC + FS.

- You need to run in circuits pre_norm.v, and show the results like next slide.
## Result

<table>
<thead>
<tr>
<th></th>
<th>Total Faults</th>
<th>Test Coverage</th>
<th># of Patterns</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC + TMAX</td>
<td>71298</td>
<td>100%</td>
<td>138</td>
<td>0.83s</td>
</tr>
<tr>
<td>DFTA + FS</td>
<td>12207</td>
<td>100%</td>
<td>201</td>
<td>0.66s</td>
</tr>
<tr>
<td>DC + FS</td>
<td>75208</td>
<td>100%</td>
<td>203</td>
<td>0.51s</td>
</tr>
</tbody>
</table>
References

- Mentor Graphic User Guide
- Synopsys TetraMax User Guide