Lab2
Scan Chain Insertion and ATPG Using DFTADVISOR and FASTSCAN

Prof: Chia-Tso Chao
TA: Yu-Teng Nien
2019-05-31
Outline

- Introduction
- DFTADVISOR
- FASTSCAN
- Mixed Flow
- Lab
Outline

- Introduction
- DFTADVISOR
- FASTSCAN
- Mixed Flow
- Lab
Introduction

- This lab focuses on ATPG with tools from 2 different EDA vendors
  - Synopsys
  - Mentor Graphics
- DFTAdvisor inserts scan chain
  - Basically replace FFs with scan FFs
- Fastscan performs ATPG and fault simulation
Insert Scan and ATPG Flow
Input/Output Files

Simulation Library
adk.atpg

DFTAdvisor

Scanned Circuit
design_scan.v

Fastscan

Scan Chain Information
- design_scan.dofile
- design_scan.testproc

Gate-Level Netlist
design.v

Simulation
Testbenches

ATE Vectors

Fault Reports
Outline

- Introduction
- DFTADVISOR
- FASTSCAN
- Mixed Flow
- Lab
Invoke DFTADVISOR

☐ Read in Verilog source file and assign ATPG library file

- $ dftadvisor pre_norm_noscan.v -verilog -lib l90sprvt.atpg -nogui

☐ Default system mode is “SETUP”

- SETUP>
Specify Clock

- Clocks are primary input signals that synchronously change the state of sequential logic elements

  - SETUP> add_clocks 0 clk

- Check the clock list

  - SETUP> report_clocks

  positive edge-triggered signal

  primary input to design
Setup Test Logic Configuration

- Set scan style design
  - Mux_scan: mux-DFF
  - Lssd: level sensitive
  - 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

- Verify with report_environment
  Non-scannable
  Scannable after test logic insertion
Enter Scan Insertion System Mode

- Enter Scan Insertion system mode (DFT) and perform scan identification
  - SETUP> set_system_mode dft

- Report detailed statistical report of scan identification
  - #Sequential instances
  - #Scannable instances
    - DFT> report_statistics
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_chains
  ▪ DFT> report_test_logic
Output Scanned Design for ATPG

- Write out files and exit DFTADVISOR
  - 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
- Mixed Flow
- Lab
Invoke Fastscan

- Specify scanned Verilog file and ATPG 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

- Enter ATPG mode
  - SETUP> set_system_mode atpg

- Select fault type: stuck, IDDQ, transition, path_delay, bridge, etc
  - ATPG> set_fault_type stuck
Generate Patterns (1/2)

- Use "-auto" option to
  - Suggest the best settings possible to generate the most compact patterns with the highest coverage within the lowest time
  - ATPG> create_patterns -auto
Without the “auto” option, you can specify your own configurations using these commands:

- `ATPG> set_atpg_limits`  
  - `cpu_seconds [integer]`  
  - `test_coverage [real]`  
  - `pattern_count [integer]`

- `ATPG> set_abort_limit [integer]`

- `ATPG> create_patterns`
During ATPG

- ATPG is performed pass after pass

<table>
<thead>
<tr>
<th>#patterns</th>
<th>test coverage</th>
<th>#faults</th>
<th>#faults detected</th>
<th># eff. patterns</th>
<th># test patterns</th>
<th>process CPU time</th>
<th>RE/AU/AAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>1.15 sec</td>
<td>0/0/15</td>
</tr>
<tr>
<td>64</td>
<td>87.16%</td>
<td>781</td>
<td>4761</td>
<td>60</td>
<td>60</td>
<td>1.23 sec</td>
<td>87/1/91</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>2.33 sec</td>
<td>117/1/103</td>
</tr>
<tr>
<td>128</td>
<td>95.93%</td>
<td>243</td>
<td>450</td>
<td>51</td>
<td>111</td>
<td>2.33 sec</td>
<td>87/1/91</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>2.58 sec</td>
<td>117/1/103</td>
</tr>
<tr>
<td>192</td>
<td>98.81%</td>
<td>70</td>
<td>143</td>
<td>50</td>
<td>161</td>
<td>2.59 sec</td>
<td>117/1/103</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>2.59 sec</td>
<td>117/1/103</td>
</tr>
<tr>
<td>229</td>
<td>99.73%</td>
<td>15</td>
<td>55</td>
<td>31</td>
<td>192</td>
<td>2.59 sec</td>
<td>117/1/103</td>
</tr>
</tbody>
</table>
ATPG Result

☐ 4 main parts
  ■ Fault number (#FU)
  ■ Test/Fault coverage
  ■ Pattern count
  ■ Runtime
☐ Print ATPG statistics report
  ■ ATPG> report_statistics

### Statistics Report

#### Stuck-at Faults

<table>
<thead>
<tr>
<th>Fault Classes</th>
<th>#faults (total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FU (full)</td>
<td>6098</td>
</tr>
<tr>
<td>UO (unobserved)</td>
<td>15 (0.25%)</td>
</tr>
<tr>
<td>DS (det_simulation)</td>
<td>5409 (88.70%)</td>
</tr>
<tr>
<td>DI (det_implication)</td>
<td>540 (8.86%)</td>
</tr>
<tr>
<td>UU (unused)</td>
<td>16 (0.26%)</td>
</tr>
<tr>
<td>RE (redundant)</td>
<td>117 (1.92%)</td>
</tr>
<tr>
<td>AU (atpg_untestable)</td>
<td>1 (0.02%)</td>
</tr>
</tbody>
</table>

#### Coverage

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>test_coverage</td>
<td>99.73%</td>
</tr>
<tr>
<td>fault_coverage</td>
<td>97.56%</td>
</tr>
<tr>
<td>atpg_effectiveness</td>
<td>99.75%</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#test_patterns</td>
<td>192</td>
</tr>
<tr>
<td>#simulated_patterns</td>
<td>229</td>
</tr>
<tr>
<td>CPU_time (secs)</td>
<td>300.5</td>
</tr>
</tbody>
</table>
View Report

- Display fault information
  - ATPG> report_faults -all
- Each fault is associated with a fault class/code

### Fault Table

```
ATPG> REPORT FAults -class ATPG_UNTESTABLE
0  AU  /IS7/OUT
1  EQ  /IS7/IN
0  EQ  /IS1/en
1  AU  /IS7/OUT
0  EQ  /IS7/IN
1  EQ  /IS1/en
0  AU  /IS4/en
0  AU  /IS20/en
1  AU  /IS20/en
1  AU  /IS2/en
```

- Fault value: Either 0 (for stuck-at-0) or 1 (for stuck-at-1)
- Fault code
- Fault site
Fault Classes - Full (FU)

- FU = TE + UT
- TE: Testable
- UT: Untestable
  - Faults which no pattern can exist to either detect or possible-detect
  - Cannot cause functional failures, so they are excluded from test coverage calculation
Fault Classes - Testable (TE)

- DT: Detected
- UD: Undetected
  - Faults that cannot be proven untestable or ATPG_untestable
  - Initial class for testable faults
- AU: Atpg_untestable
  - Due to pin constraint or insufficient sequential depth placed on Fastscan
- PD: Possible-detected
  - Faults with good-machine value being either 0 or 1 and faulty machine value being X in simulation
Fault Classes - Untestable (UT)

- **UU: Unused**
  - Faults not connected to any circuit observation point

- **BL: Blocked**
  - Faults blocked by logic on all paths

- **TI: Tied**
  - Point of the fault value is always same (e.g. AND2 with complementary inputs)

- **RE: Redundant**
  - Faults undetectable after exhausting all patterns and need dedicated analysis to verify redundancy
  - **ATPG> identify_redundant_faults**
Test Coverage Formula Comparison

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

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

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

- **Fastscan**
  
  \[
  \text{test\_coverage} = \frac{DT + (PD \times \text{posdet\_credit})}{\text{testable}} \times 100
  \]

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

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

\[\text{Testable} = DT + PD + AU + UD \quad \text{Untestable} = UU + TI + BL + RE\]
Save Patterns

- Save patterns that are generated via ATPG
- Various formats 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 -replace
  - ATPG> exit

Toshiba Standard Tester Interface Language 2
Mixed Flow

- Synopsys Design Compiler is way better at mapping from RTL code to gate-level netlist
- Some practices in industrial project hence adopt
  - Design Compiler to synthesize gate-level netlist and do scan-chain insertion
  - Fastscan to perform ATPG
Input/Output Files

- Simulation Library
  - tsm18.v

- DC

- Scanned Circuit
  - design_scan.v

- Scan Chain Information
  - design_scan.stil

- Fastscan
  - stil2mgc

- Simulation Testbenches
- ATE Vectors
- Fault Reports
Input Files Required in Mixed Flow

- Library file needs to be converted
  - From .v to .atpg
- The detailed information as to scan chain needs to be converted
  - From .stil to .dofile
- Scanned design
Convert Library File

- 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.atpg -rep
  - TRANSLATION> exit
Convert STIL File

- Use `stil2mgc` to convert STIL file from Design Compiler into Fastscan-compatible dofile and test procedure file
  - `$ stil2mgc pre_norm_scan.stil`
  - It generates both files
    - `pre_norm_scan.stil.do`
    - `pre_norm_scan.stil.proc`
Perform ATPG using FASTSCAN

☐ Read scanned circuit 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_system_mode atpg
- ATPG> create_patterns -auto
- ATPG> report_statistics
Outline

- Introduction
- DFTADVISOR
- FASTSCAN
- Mixed Flow
- Lab
Lab Goal

- Compare the following during ATPG using the DC+TMAX, DFTA+FS and DC+FS flows
  - Total fault number
  - Test coverage
  - Pattern count
  - Run time (s)
- Run on circuit “pre_norm_noscan.v”, and show a table like next slide
# Example of Lab Result

<table>
<thead>
<tr>
<th>Flow</th>
<th>#Faults</th>
<th>Test Coverage</th>
<th>#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>DFT+FS</td>
<td>122072</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 Graphics
  - DFTAdvisor Reference Manual, v8.6_4
  - Tessent Scan and ATPG User’s Manual, v2014.1

- Synopsys
  - TetraMAX ATPG User Guide, J-2014.09-SP1