DFT Compiler & TetraMAX

Kate, Yu-Jen Huang

Dec 17 2009
Outline

- VLSI Testing
  - Introduction
  - Fault modeling
  - Test generation
- Design for Testability (DFT)
- Fault Simulation (TetraMAX)
- Lab time
Definitions

☐ Design synthesis
  ▪ Give an I/O function, develop a procedure to manufacture a device using known materials and processes

☐ Verification
  ▪ Predictive analysis to ensure that the synthesized design will perform the given I/O function

☐ Testing
  ▪ A manufacturing step that ensures that the physical device, manufactured from the synthesized design, has no manufacturing defect
Why Testing?

- Economy
  - Product quality
  - Product reliability

- The purpose of testing is to weed out all bad products before they are shipped to users
  - The number of bad products heavily affect the price of good products

- A profound understanding of the principles of manufacturing and test is essential for a designer to design a quality product
Defect, Fault, and Error

- Defect
  - A defect is an unintended difference between the implemented hardware and its intended design
  - It is caused during manufacture or the use of devices

- Fault
  - A representation of a physical defect at the abstracted function level

- Error
  - A wrong output signal produced by a defective circuit
  - It is caused by a fault or a design error
Testing Problem

- What faults to test
  - Fault modeling
- How are test pattern obtained
  - Test pattern generation
- How is test quality
  - Fault simulation
- How are test vectors applied and results evaluated
  - ATE/BIST
Modern IC Testing

Test program

Devise under test (DUT) -> Automatic test equipment (ATE) -> Passed the test

Failed the test

Advanced Reliable Systems (ARES) Lab.
Yu-Jen Huang
Cost of Test

- **Design for testability (DFT)**
  - Area overhead and yield reduction
  - Performance overhead

- **Software processes of test**
  - Test generation and fault simulation
  - Test programming and debugging

- **Manufacturing test**
  - Automatic test equipment (ATE) capital cost
  - Test center operational cost
Outline

- VLSI Testing
  - Introduction
  - Fault modeling
  - Test generation
- Design for Testability (DFT)
- Fault Simulation (TetraMAX)
- Lab time
Real defects are too numerous and hard to be analyzed
Fault

- Fault is a physical defect in a circuit/system
  - Permanent Fault: a fault that is continuous and stable, whose nature do not change before, during, and after testing
    - Hard fault or solid fault
  - Temporary fault: a fault that is present only part of the time, occurring at random moments and affecting the system for finite, but unknown, intervals of time
    - Transient fault, soft error
  - Intermittent fault: caused by non-environmental conditions
Fault Model and Error

- Fault model is a logic effect of a fault
  - Structural fault
    - Stuck-at-faults
    - Bridging fault
    - Open fault
    - Transition fault
    - Delay fault
  - Functional fault
    - RAM coupling and pattern-sensitive faults

- Error is manifestation of a fault that results in an incorrect module output or system state
Failure

- Failure is deviation of a system from its specified behavior
  - Fault -> error -> failure

- Failure mechanism is a physical or chemical process that causes devices to malfunction

- Failure mode is the cause of rejection of failed device (effect of failure mechanism), such as open/short interconnections, or degraded parameter values
Defect Level, Fault Coverage and Yield

- Defect Level
  - The fraction of devices that pass all the tests but still contain faults
  - $DL = 1 - Y^{(1-FC)}$

- Fault Coverage (FC)
  - The measure of the ability of a test set $T$ to detect a given set of faults
  - $FC = \frac{\text{No. of detected faults}}{\text{No. of possible faults}}$
  - Can be determined by fault simulation

- Yield ($Y$) = No. of good dies per wafer/No. of dies per wafer
Defect Level and Fault Coverage

- DL is measured in terms of DPM (defects per million), and typical values claimed are less than 200DPM, or 0.02%
- Required FC for DL = 200DPM

<table>
<thead>
<tr>
<th>Y(%)</th>
<th>10</th>
<th>50</th>
<th>90</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC(%)</td>
<td>99.99</td>
<td>99.97</td>
<td>99.8</td>
<td>98</td>
</tr>
</tbody>
</table>
Stuck-at-fault

- Single stuck-at-fault: line has a constant value
- Multiple stuck-at-fault: several single SAFs occur at the same time

![Logic Diagram]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>c(a/0)</th>
<th>c(a/1)</th>
<th>c(b/0)</th>
<th>c(b/1)</th>
<th>c(c/0)</th>
<th>c(c/1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Single Stuck-at Fault Assumption

- ATPG tools assume at most one SA fault is present on the chip under test.
- This assumption is made to simplify the analysis.
  - Detecting multiple faults would complicate ATPG.
- The SSF model disregards the possible presence of any other faults affecting the test for a target fault.
- SSF assumes no chance of another fault masking the target fault, making it impossible to detect.
Number of Single Stuck-at Faults

- Number of faults in a Boolean gate circuit
  - No. of PI + No. of gates + No. of fanout branches
- Example: XOR gate
  - 24 SAFs
Bridging Faults

- Two or more normally distinct points (lines) are shorted together

- Two types of bridging faults
  - Input bridging
    - Can form wired logic or voting model
  - Feedback bridging
    - Can introduce feedback
Bridging Fault behavior

- **AND bridges**
  - The bridge acts like an AND gate

- **OR bridges**
  - The bridge acts like an OR gate

- **Dominant bridges**
  - The dominating net always controls the value output
Single Cell Fault

- Cells can have any implementation
- All possible cell faults are allowed
- C-testability: constant number of test patterns, independent of circuit size

---

Advanced Reliable Systems (ARES) Lab.  
Yu-Jen Huang
Transistor Faults

- MOS transistor is considered an ideal switch and two types of faults are modeled
  - Stuck-open: a single transistor is permanently stuck in the open state
  - Stuck-on: a single transistor is permanently shorted irrespective of its gate voltage
- Detection of a stuck-open fault requires two vectors
- Detection of a stuck-on fault requires the measurement of quiescent current ($I_{DDQ}$)
Timing-Related or Delay Fault

- Delay fault
  - Gate delay fault: A gate delay fault occurs when a gate operates more slowly than expected

- Path delay fault: A path delay fault assumes that a logic transition is delayed along an entire path

- Because delays refers to differences in behavior over time, delay faults focus on transition in logic values
Outline

- VLSI Testing
  - Introduction
  - Fault modeling
  - Test generation
- Design for Testability (DFT)
- Fault Simulation (TetraMAX)
- Lab time
Testing

- Testing = test generation + test application + output evaluation
- FC can be determined by fault simulation
- Cost of test generation (TG) depends on
  - Complexity of the fault model
  - Complexity of the TG algorithm
  - Complexity of the DUT
- A test set for a class of faults $F$ is a set of tests $T$ such that for any fault $f \in F$, there exists $t \in T$ such that $t$ detects $f$
**Test Generation by Truth Table**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>c(a/0)</th>
<th>c(a/1)</th>
<th>c(b/0)</th>
<th>c(b/1)</th>
<th>c(c/0)</th>
<th>c(c/1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- $T_{a/0} = \{11\}$; $T_{a/1} = \{01\}$; $T_{b/0} = \{11\}$; $T_{b/1} = \{10\}$
- $T_{c/0} = \{11\}$; $T_{c/1} = \{00\}$ or $\{01\}$ or $\{10\}$
- $T = \{01, 10, 11\}$
Undetectable Fault

- No pattern can be devised to detect fault U2 SA0
D-algorithm

- Select a primitive cube to activate fault f
- Sensitize all possible paths from the fault site to POs (fault propagation or D-drive)
  - Continued until a PO has a D or D’
- Develop a consistent set of primary input (PI) values that will account for all lines set to 0 or 1 during D-drive. If not consistent, try another path
Example

Advanced Reliable Systems (ARES) Lab.

Yu-Jen Huang
Outline

- VLSI Testing
  - Introduction
  - Fault modeling
  - Test generation
- Design for Testability (DFT)
- Fault Simulation (TetraMAX)
- Lab time
Design for Testability (DFT)

- A fault is testable if there is a well-specified procedure to expose it, which can be implemented with a reasonable cost using current technique.

- DFT
  - A class of design methodologies which put constraints on the design process to make test generation and diagnosis easier.
Sequential Logic - I

Advanced Reliable Systems (ARES) Lab.  Yu-Jen Huang
Sequential Logic - II

- Harder to test
  - Sequential circuit has memory in addition to combinational logic
  - It takes more clock cycles to activate the fault and propagate the fault effect

- Example

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```

```
Advanced Reliable Systems (ARES) Lab.  Yu-Jen Huang
```
Scan Design Approaches

- **MUX scan**
  - Shift register modification

- **Clock scan**
  - LSSD

- **Partial scan**

- **Boundary scan**
  - 1149.1 (JTAG)
  - 1149.4
  - 1149.5
Mux Scan

Primary Inputs (PIs) → Combinational Logic → Primary Outputs (POs)

Primary inputs → Combinational Logic → Primary outputs

Pseudo Primary Outputs (PPOs) → Sequential Logic → Scan output (SO)

Pseudo Primary inputs (PPIs) → Sequential Logic → Scan input (SI)

Controllability and observability

Advanced Reliable Systems (ARES) Lab.
Yu-Jen Huang
Scan Cell

Combinational Logic

PI  PPO  PPI  mux  SI  T  clk  .....  SO  PO

Advanced Reliable Systems (ARES) Lab.  Yu-Jen Huang
Sequential Logic – Full Scan

Scan_en

Scan_in

clk

Scan_out

Advanced Reliable Systems (ARES) Lab.

Yu-Jen Huang
Operation

- 1. Switch to SR mode
- 2. Initialize SR
  - Load the first pattern
- 3. Return to normal mode
  - Apply test pattern
- 4. Switch to SR mode
  - Shift out the final state
  - Setting the starting state for the next test
  - Go back to the previous operation 3
Advantages and Disadvantages

Advantages

- Scan paths allow use of combinational ATPG
- Make high coverage possible in reasonable ATPG time and test program length

Disadvantages

- Area overhead
  - Scan-in pin (SI), scan-out pin (SO), test mode pin (T), and mux in front of each FF
- For large design, back-end re-optimization to fix scan timing is tedious and time-consuming
Scan Synthesis

- Scan configuration
  - Number of scan chains
  - Types of scan cells
  - Storage elements to exclude from scan synthesis
  - How scan cells are arranged within scan chains

- Scan replacement
  - Replace original design to scannable design

- Scan reordering
  - Reorder scan chains

- Scan stitching
  - Connect all scan cells together to form scan chains
1. Before DFT Synthesis

- Add scan related I/O pins in the design
  - Add `si`, `se`, `testmode` as the input pins
  - Add `so` as the output pin

```vhdl
module cpu(
  clk,       // system clock
  rst,       // system reset
  control,   // instruction input
  memaddr,   // address of RAM
  in,        // data input
  out,       // data output
  si,        // scan in
  se,        // scan enable
  so,        // scan out
  test_mode  // test mode
);
```

```
// 92521013@cc.ncu.edu.tw (Tsu-Wei Tseng)
//Date  : 2008/08
//Abstract: This CPU supports 16 instructions. Also, a 64-byte RAM, an
// ALU and 3 data registers are included

//include "RAM_64B.v"
```

Advanced Reliable Systems (ARES) Lab.
Yu-Jen Huang
2. Set Scan Chain Type

☐ Set in .synopsys_dc.setup file
  ■ set_scan_configuration -style multiplexed_flip_flop
DFT Compiler Flow

Set constraints: scan style, speed, area

Constraint-based scan synthesis: Routing, balancing, gate-level opt.

Preview coverage
Scan Synthesis Flow

1. Create Test Protocol
2. DFT Check
3. Test-Ready Compile
4. Design + Test Protocol
5. Specify Scan Chain
6. DRC & Preview
7. Scan Chain Synthesis
8. Scan Chain Identification
9. DRC & Coverage
10. Handoff Design

Existing Scan Chain Flow

Top-Level Netlist

Advanced Reliable Systems (ARES) Lab.
Yu-Jen Huang
3. Create Test Protocol

- Set_dft_signal
  - `set test_default_period 100`
  - `set_dft_signal -view existing_dft -type ScanClock -timing {45 55} -port clk`
  - `set_dft_signal -view existing -type Reset -active_state 1 -port rst`
  - `set_dft_signal -view existing -type Constant -active_state 1 -port test_mode`
  - `create_test_protocol`
4. DFT Check

☐ Pre-DFT DRC
☐ Check scan design rule before scan chain synthesis
☐ dft_drc
5. Test-Ready Synthesis

- compile -scan

![Diagram showing DFF with inputs D, clk, and Q, replaced by a DFF with inputs D, se, clk, and Q.]
6. Read Design & Test Protocol

- Write out the test protocol and scan-ready design
  - `write_test_protocol -output cpu.spf`
  - `Write -format ddc -hierarchy -output cpu.ddc`

- Read design & test protocol
  - `read_file -format ddc cpu.ddc`
  - `current_design design`
  - `link`
  - `read_test_protocol cpu.spf`
Set global attributes for scan paths in the current design

- `set_scan_configuration -chain_count 1`
- `set_scan_configuration -clock_mixing no_mix`
- `set_dft_signal -view spec -type ScanDataIn -port si`
- `set_dft_signal -view spec -type ScanDataOut -port so`
- `set_dft_signal -view spec -type ScanEnable -port se -active_state 1`
- `set_scan_path chain1 -scan_data_in si -scan_data_out so`
Memory Block

Advanced Reliable Systems (ARES) Lab.  Yu-Jen Huang
Add in step 5

- `set_test_point_element -type observe [get_object_name [get_pins RAM_64B/D*]] -clock_signal clk`
- `set_test_point_element -type observe [get_object_name [get_pins RAM_64B/A*]] -clock_signal clk`
- `set_test_point_element -type control_01 [get_object_name [get_pins RAM_64B/Q*]] -clock_signal clk`
- `report_test_point_element`
8. Scan Preview

- Check scan-path consistency
- Determines the chain count
- Allocates and orders scan cells
- Adds connecting hardware
  - `preview_dft -show all`
  - `preview_dft -test_points all`
9. Scan Chain Synthesis

- Scan replacement
- Ensures no contention
- Inserts test points
- Optimized the logic
  - insert_dft
10. Scan Chain Identification

☐ Use when you import an existing scan design in non-ddc netlist format

- `set_scan_state scan_existing`
11. DRC & Coverage

- Post-DFT DRC
  - `dft_drc -coverage_estimate`

---

Resetting current test mode
Beginning Mapping Optimizations

In mode: Internal_scan...
Design has scan chains in this mode
Design is scan routed
Post-DFT DRC enabled

Information: Starting test design rule checking. (TEST-212)
Loading test protocol
...basic checks...
...basic sequential cell checks...
...checking vector rules...
...checking clock rules...
...checking scan chain rules...
...checking scan compression rules...
...checking X-state rules...
...checking tristate rules...
...extracting scan details...
...saving simulation value info...

Begin Modeling violations...

---

Pattern Summary Report

| $internal patterns | 0 |

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>2164</td>
</tr>
<tr>
<td>Possibly detected</td>
<td>PT</td>
<td>0</td>
</tr>
<tr>
<td>Undetectable</td>
<td>UD</td>
<td>90</td>
</tr>
<tr>
<td>ATPG untestable</td>
<td>AU</td>
<td>150</td>
</tr>
<tr>
<td>Not detected</td>
<td>ND</td>
<td>0</td>
</tr>
</tbody>
</table>

| total faults | 2404 |
| test coverage | 93.52% |

Information: The test coverage above may be inferior than the real test coverage with customized protocol and test simulation library.

---

Advanced Reliable Systems (ARES) Lab.  Yu-Jen Huang
12. Handoff Design

- Report scan information
  - `report_scan_path -view existing_dft -chain all`
  - `report_scan_path -view existing_dft -cell all`

---

<table>
<thead>
<tr>
<th>Scan_path</th>
<th>Cell_#</th>
<th>Instance_name</th>
<th>Clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>I chain</td>
<td>0</td>
<td>reg_a_reg[0]</td>
<td>clk</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>reg_a_reg[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>reg_a_reg[2]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>reg_a_reg[3]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>reg_a_reg[4]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>reg_a_reg[5]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>reg_a_reg[6]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>reg_a_reg[7]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>regb_reg[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>regb_reg[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>regb_reg[2]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>regb_reg[3]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>regb_reg[4]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>regb_reg[5]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>regb_reg[6]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>regb_reg[7]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>regc_reg[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>regc_reg[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>regc_reg[2]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>regc_reg[3]</td>
<td></td>
</tr>
</tbody>
</table>
13. Handoff Design

☐ Prepare TetraMax script

- `change_names -hierarchy -rule verilog`
- `write -format verilog -hierarchy -out cpu_dft.vg`
- `write -format ddc -hierarchy -output cpu_dft.ddc`
- `write_scan_def -output cpu_scan.def`
- `set test_stil_netlist_format verilog`
- `write_test_protocol -output cpu.spf`
Outline

- VLSI Testing
  - Introduction
  - Fault modeling
  - Test generation for combinational and sequential circuits
- Design for Testability (DFT)
- Fault Simulation (TetraMAX)
- Lab time
**Automatic Test Pattern Generation**

- **Goal**
  - Generate the test patterns for target fault model and keep the number of test pattern as small as possible

- **How?**
  - Use an ATPG tool which relies on proprietary techniques to speed up and extend the basic D algorithm

---

**Flowchart**

1. **Fault list**
   - Remove all detected fault and select next fault

2. **Test Generation**
   - Add pattern to test set

3. **Fault simulation**

---

_Note:

Advanced Reliable Systems (ARES) Lab.
Yu-Jen Huang_
DFT Compiler to TetraMAX

Design Compiler

Simulation Library

TetraMax

- read netlist design_dft.v
- read netlist library.v

- write –f verilog –hierarchy -output “design_dft.v”
- write_test_protocol –f stil -out “design.spf”
- run drc design.spf

Simulation Testbenches

ATE Vectors

Fault Reports

Advanced Reliable Systems (ARES) Lab.  Yu-Jen Huang
5 Fault Categories

During ATPG and functional fault simulation, TetraMAX classifies faults into 5 major categories:

- DT: Detected
- PT: Possibly detected
- UD: Undetected
- AU: ATPG untestable
- ND: Not detected

Test coverage

\[
\frac{\text{DT} + (\text{PT} \times \text{posdet_credit})}{\text{all faults} - (\text{UD} + (\text{AU} \times \text{au_credit}))}
\]
TetraMAX

- Setup file (CSHRC)
  - `source /usr/cad/synopsys/CIC/tmax.csh`
  - `source /APP/cshbank/tmax.csh`

- Invoking TetraMAX
  - `tmax&`

Advanced Reliable Systems (ARES) Lab.

Yu-Jen Huang
Command Input and Script Files

- Command input may be specified in many ways:
  - manus, GUI buttons, dialog boxes
  - typed at the command input line
  - read from command files

- By default, TetraMAX aborts a script file when a command returns an error.

- To continue executing scripts, use:
  - BUILD> set_command noabort
Multi-line Input in the GUI

- Multiple commands may also be entered by separating each command by a semicolon and white space
  - BUILD> build –force; read net mylib.v; run build
Help Command

- **BUILD> help add**
  
  Add Atpg Constraints   Add Atpg Gates
  Add Cell Constraints   Add Clocks
  Add Equivalent Nofaults Add Faults
  Add Net Connections    Add Nofaults
  Add PI Constraints     Add PI Equivalences
  Add PO Masks

- **BUILD> help read netlist**
  
MAN for Command Reference

- Entering “man” and a command name, a message ID, or a DFT rule ID or violation ID will open up the on-line help to the reference page for that topic.

BUILD> man getting_started // a topic
BUILD> man add clock // a command
BUILD> man report faults // a command
BUILD> man z4-6 // a violation
BUILD> man m68 // a message ID
Stop Process

- Submit button changes to “Stop” while performing operation
- or CTRL+C and CTRL+Break
Basic Flow

- Read in the netlist
- Build Mode
  - BUILD> read netlist xxx.vg
  - BUILD> read netlist tsmc18.v
  - BUILD> run build_model top_model
- DRC mode
  - DRC> set drc xxx.spf
  - DRC> run drc
1. Build Mode - Read Netlist

1. Read cpu_dft.vg & tsmc18.v & RAM_64B.tv

2. Run

Advanced Reliable Systems (ARES) Lab.
Yu-Jen Huang
2. Choose Top Module

1. Build

2. Choose top module "cpu"

3. Run

Advanced Reliable Systems (ARES) Lab.

Yu-Jen Huang
3. DRC Mode – Choose spf file

1. DRC

2. Choose spf file “cpu_dft,spf”

3. Run
4. Test Mode

- Test mode
  - TEST> report summaries faults patterns
  - TEST> add faults -all
  - TEST> report summaries faults patterns
  - TEST> run atpg -auto
  - TEST> report summaries faults patterns

- Increase fault coverage
  - run atpg -fast seq. (or full seq.)

- Write ATPG patterns
  - write patterns xxx_atpg.v -internal -format verilog -serial -replace
Test Coverage

Advanced Reliable Systems (ARES) Lab.

Yu-Jen Huang
Reference

☐ DFT compiler user guide
☐ TetraMax user guide
LAB TIME