Lab1
Scan-Chain Insertion And ATPG

Pro: Chia-Tso Chao
TA: Yu Pang, Hu
2018/05/28

Outline

- Introduction
- Design Compiler
- TetraMax
- Lab

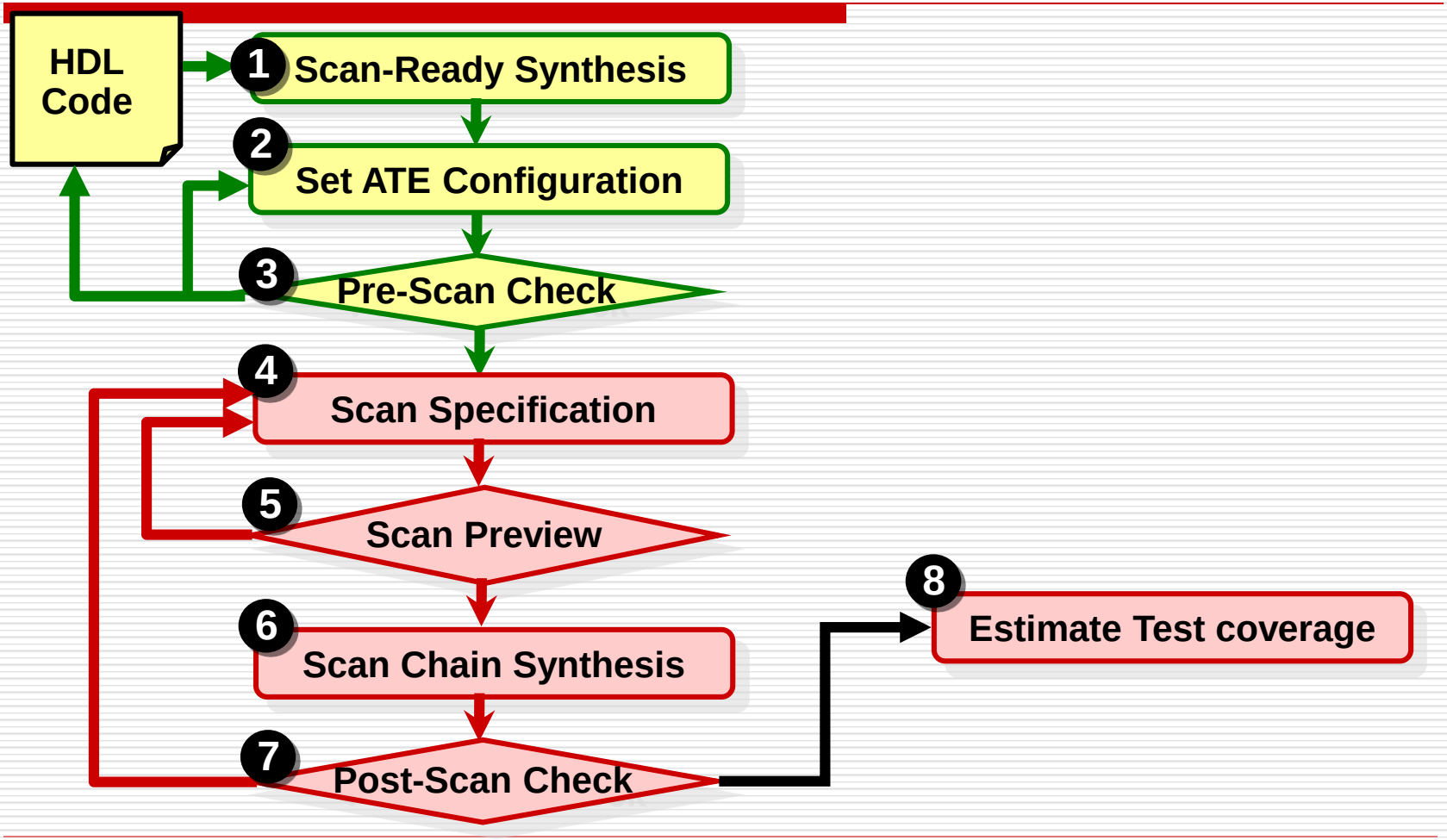
Outline

- Introduction
- Design Compiler
- TetraMax
- Lab

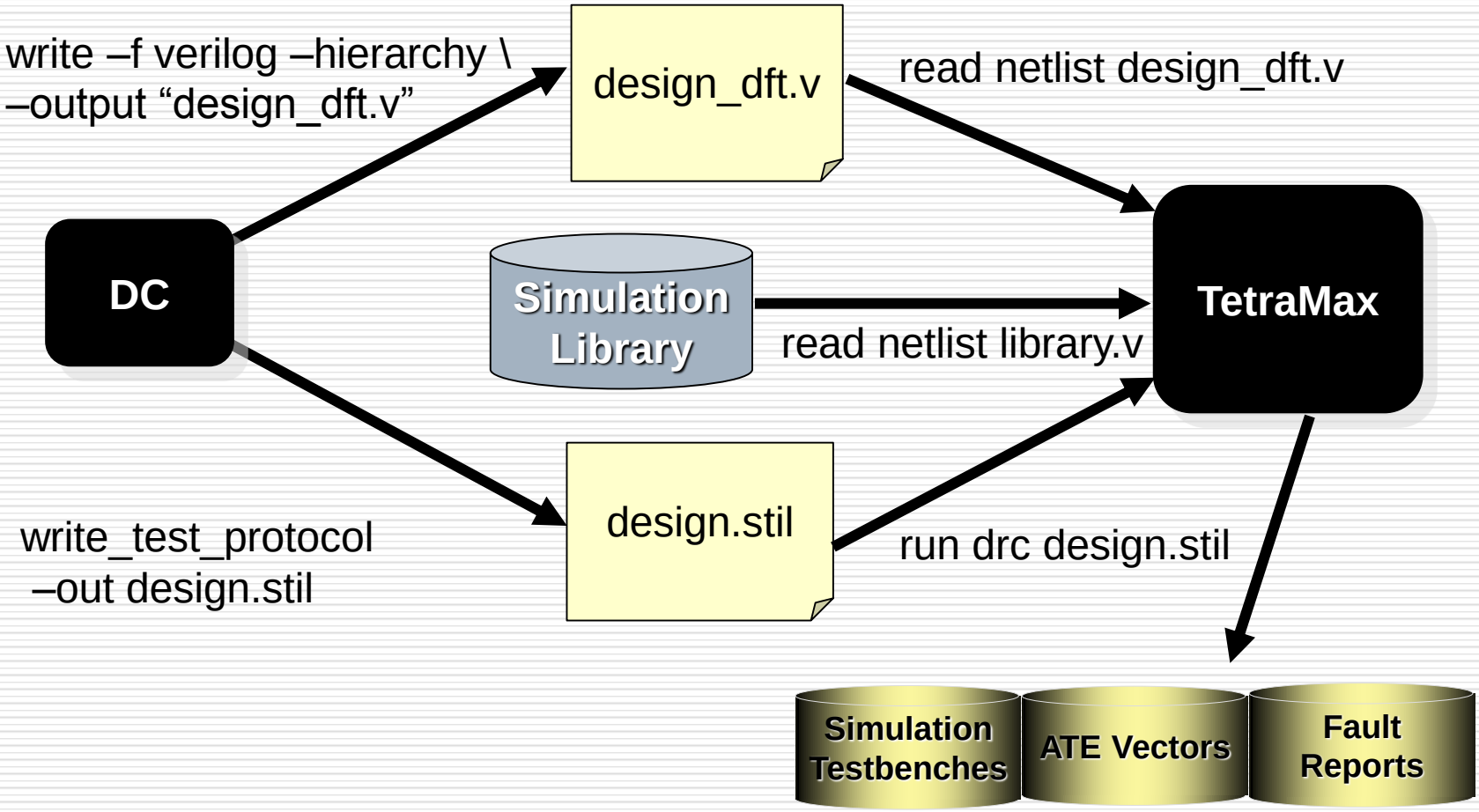
Introduction

- This lab compares impact on circuit after scan-chain insertion.
- Items being compared including area, power, test coverage, # of patterns.
- Synopsys Design Compiler is the most common synthesis tool supports interactive command input.
- Synopsys TetraMax is used to perform ATPG (Automatic Test Pattern Generation) and fault simulation.

Scan Synthesis Flow



DFT compiler to TetraMax



Outline

- Introduction
- Design Compiler
- TetraMax
- Lab

Work Stations

- Open the terminal
- $ ssh [Account]@linux27.ee.nctu.edu.tw
  - Account: vlsitest01~vlsitest12
  - Password: vlsitest

Setup Tool Environment

- Defined in .synopsys_dc.setup (it's a hidden file, so use command $ ls –al to find it)
  - set link_library "l90sprvt_typ.db"
  - set target_library "l90sprvt_typ.db"

  - set hdlin_translate_off_skip_text "TRUE"
  - set edifout_netlist_only "TRUE"
  - set verilogout_no_tri true
  - set plot_command {lpr -Plp}

Invoke Design Compiler

- cd lab1
- $ tcsh
- $ dc_shell-t

Read File, Link, Uniquify

- Read in RTL verilog source files
  - dc_shell> read_file -format verilog pre_norm.v
- Show library details
  - dc_shell> list_libs
- Specify the current module to sythesize
  - dc_shell> current_design pre_norm
    - pre_norm : top module
- Link
  - Resolve the design reference based on reference names
  - Locate all design and library components, and connect them
    - dc_shell> link
- Uniquify
  - Removes multiply-instantiated hierarchy in the current design by creating a unique design for each cell instance
    - dc_shell> uniquify

Wire Model, Scan Style, Clock

- Setup wire load model define in library
  - dc_shell> set_wire_load_model -name wl10 -library l90sprvt_typ
    - Use 'report_lib l90sprvt_typ' to view library information
- Specify the scan style. Four styles are supported
  - 1) Multiplexed flip-flop (multiplexed_flip_flop)
  - 2) Clocked scan (clocked_scan)
  - 3) Level-sensitive scan design (lssd)
  - 4) Auxiliary-clock LSSD (aux_clock_lssd)
    - dc_shell> set_scan_configuration -style multiplexed_flip_flop
- Specify clock
  - dc_shell> create_clock clk -period 10
    - clk : the signal name define in the HDL file

Compile(1/2)

- Using command “compile” to perform logic level and gate level synthesis and optimization on current design
  - “-map_effort” : specify the relative amount of CPU time spent during the mapping phase of compile

Compile(2/2)

- “-scan” : specify command to consider the impact of scan insertion on mission mode constraints during optimization. This option causes the command to replace all sequential elements during optimization. Some scan-replaced sequential cells might be converted to nonscan cells later in the test synthesis process because of test design rule violations or explicit user specifications.
  - dc_shell> compile -scan -map_effort medium

Identify Scan-Chain Count, Generate Test Protocol(1/3)

- Set scan-chain count considering the limitation of ATE or software, multiple clock domain, test time limitation
  - dc_shell> set_scan_configuration -chain_count 10
- Define clocks in your design, then generate a test protocol
  - infer_clock option to find clock signal
  - infer_async option to find reset signal
  - dc_shell> create_test_protocol -infer_clock -infer_async

Identify Scan-Chain Count, Generate Test Protocol(2/3)

- If you want to specify some PI/POs to be normal inputs at operation mode and scan inputs during test mode use following commands
  - dc_shell> set_scan_configuration -chain_count 1
  - dc_shell> set_dft_signal -port add -type scandatain
  - dc_shell> set_dft_signal -port sign -type scandataout
  - dc_shell> create_test_protocol -infer_clock -infer_asynch

Identify Scan-Chain Count, Generate Test Protocol(3/3)

- If you want to specify scan-chain order, user the following command
  - dc_shell> set_scan_configuration -chain_count 1
  - dc_shell> set_scan_path ch1 -ordered_elements { DFF_1 DFF_2 … DFF_50 } -complete true
  - dc>shell> create_test_protocol -infer_clock -infer_async
  - complete option to indicates whether DFT Compiler can add components to a specified scan chain.

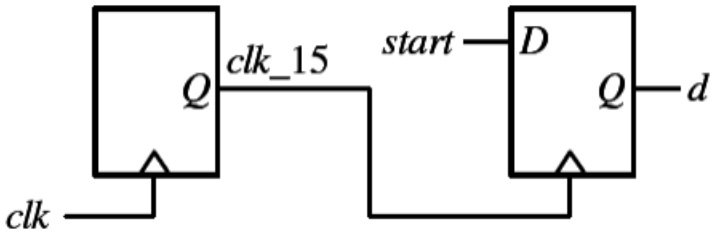
Preview Design, Scan-Chain Synthesis

- Preview the scan design
  - dc_shell> preview_dft
- Check test design rules according to the scan style you chose
  - dc_shell> dft_drc
- Insert scan chain
  - dc_shell> insert_dft

If clock is gated(DRC violation)

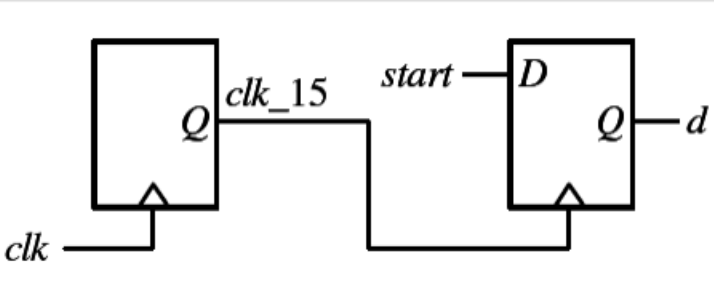
- ---
- DRC Report
- Total violations: 1
- ---
- 1 PRE-DFT VIOLATION
- 1 Uncontrollable clock input of flip-flop violation (D1)

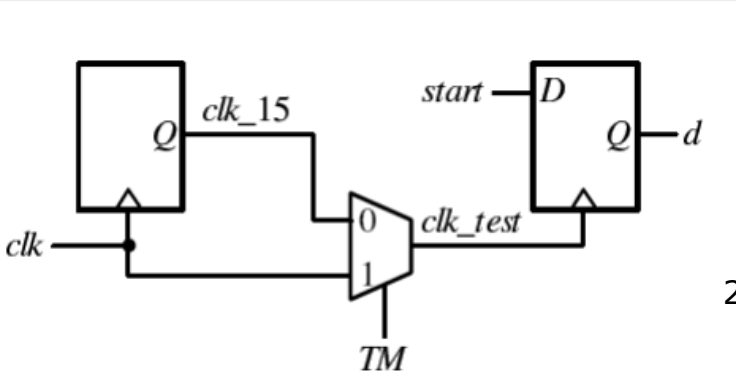
- Warning: Violations occurred during test design rule checking. (TEST-124)
- ---
- Sequential Cell Report
- 1 out of 71 sequential cells have violations
- ---
- SEQUENTIAL CELLS WITH VIOLATIONS
- * 1 cell has test design rule violations
- SEQUENTIAL CELLS WITHOUT VIOLATIONS
- * 70 cells are valid scan cells



If clock is gated(DRC violation)

- Add additional signal TM for testability
  - dc_shell> create_port -direction "in" {TM}
  - dc_shell> set_dft_configuration -fix_clock enable
  - dc_shell> set_dft_signal -view exist -type ScanClock -timing {50 100} –port clk
  - dc_shell> set_dft_signal -view spec -type TestData -port clk
  - dc_shell> set_dft_signal -view spec -type TestMode -port TM
  - dc_shell> set_autofix_configuration -type clock -control TM -test_data clk





Report Area, Time, and Power

- Report area, timing, and power
  - dc_shell> report_area
  - dc_shell> report_timing
  - dc_shell> report_power

Result(1/2)

- Area
  - Number of ports: 147
  - Number of nets: 594
  - Number of cells: 474
  - Number of references: 52

  - Combinational area: 2765.914043
  - Noncombinational area: 1302.566048
  - Net Interconnect area: 103180.518768

  - Total cell area: 4068.480091
  - Total area: 107248.998859

Result(2/2)

- Power
  - Global Operating Voltage = 1
  - Power-specific unit information :
  - Voltage Units = 1V
  - Capacitance Units = 1.000000pf
  - Time Units = 1ns
  - Dynamic Power Units = 1mW (derived from V,C,T units)
  - Leakage Power Units = 1uW

  - Cell Internal Power = 92.3638 uW (38%)
  - Net Switching Power = 151.7164 uW (62%)
  - ---------
  - Total Dynamic Power = 244.0803 uW (100%)

  - Cell Leakage Power = 4.9543 uW

Post Scan Check, Report Scan Path

- Recheck a design against the design rules of a chosen scan style
  - dc_shell> dft_drc
- Report the configuration of scan paths
  - dc_shell> report_scan_path

Write Out Synthesized Verilog And STL Files

- Save the scaned gate level netlist
  - dc_shell> write -hierarchy -format verilog -output pre_norm_scan.v
  - dc_shell> write_test_protocol -output pre_norm_scan.stil
  - dc_shell> write_sdc pre_norm_scan.sdc
  - dc_shell> write_scan_def -output pre_norm_scan.def
  - dc_shell> exit

Outline

- Introduction
- Design Compiler
- TetraMax
- Lab

Invoke TetraMax

- In the tcsh shell

- $ tmax -s

Read Netlist And Library

- Read verilog netlist file
  - BUILD> read_netlist pre_norm_scan.v
- Read library file
  - BUILD> read_netlist l90sprvt.v -library

Reporting Modules

- "-summary" : generate a summary report on all modules
- "-error" : report all modules that have at least one violation of a rule of severity of "error"
- "-undefined" : report all modules that are referenced but not defined
  - BUILD> report_modules -summary
  - BUILD> report_modules -error
  - BUILD> report_modules -undefined

Building ATPG Design Model

- Builds the in-memory simulation model from the design modules that have been read in
  - BUILD> run_build_model pre_norm
    - It will change into DRC command mode

Set DRC Parameters And Run

- Set the parameters that control DRC process. You can display the current settings with "report_settings" commands
- Perform Design Rule Checking, which is required to enter the TEST command mode, where test generation and fault simulation may be performed
  - DRC> run_drc pre_norm_scan.stil
    - pre_norm_scan.stil : scan chain configuration file

ATPG(1/4)

- Select the fault model for ATPG
  - TEST > set_faults –model stuck

- TetraMAX supports test pattern generation for five types of fault models:
  - Stuck-At
  - Transition
  - Path Delay
  - IDDQ
  - Bridging

ATPG(2/4)

- Create a list of faults for fault simulation and test generation.
  - TEST > add_faults -all
- Set the parameters that control the ATPG processes
  - "-merge" : Specify whether to perform pattern merging during ATPG. The arguments indicates how much effort to spend doing merging (default: none)
  - "-verbose" : With -verbose enabled, extra messages are displayed during the pattern merge operation

ATPG(3/4)

- "-abort_limit" : Specify the max. number of remade decisions before terminating a test generation effort during ATPG. (default: 10)
- "-coverage" : Specify a test coverage limit at which to terminate the ATPG effort. Ranging from 0 ~ 100 (default: 100)
- "-decision" : When backtracking, using specific way to determine (default: norandom)
  - TEST> set_atpg -merge high -verbose -abort_limit 250 -coverage 100 -decision random
  - TEST> run_atpg

ATPG(4/4)

- "-time" : Specify the maximum CPU time, in seconds, allowed per fault or per run. The time limit can be turned off again by specifying a 0 for the time values.
  - **Note for Tcl mode** : Multiple values specified by the -time option must appear as a list and be enclosed by curly braces "{}".
- "-full_seq_time" : Similar to "-time" option, but applies to the Full-Sequential ATPG algorithm. (default: 0)
  - TEST> set_atpg -merge high -verbose -full_seq_time {3600 86400} -full_seq_atpg
  - TEST> run_atpg

Result

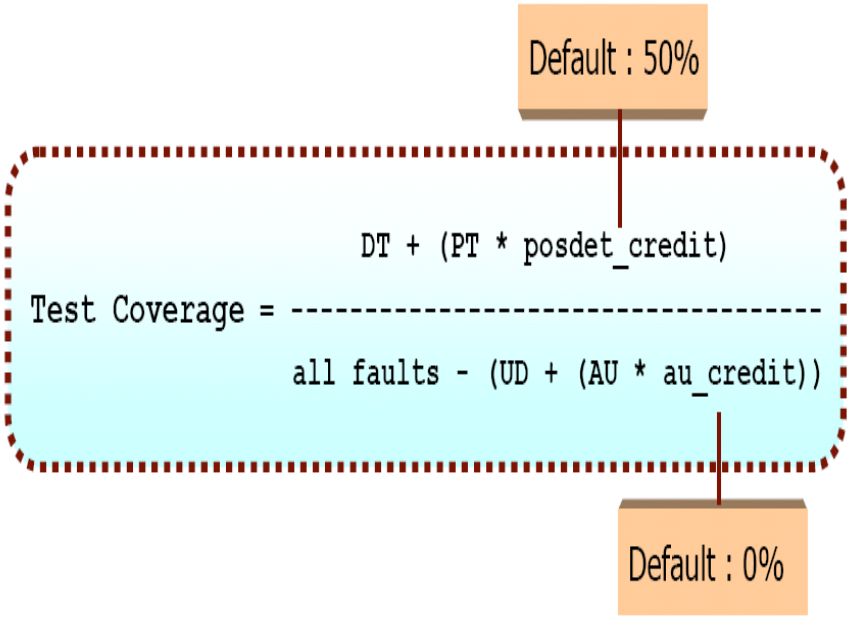
- ATPG performed for stuck fault model using internal pattern source.
- ----------
- #patterns #faults #ATPG faults test process
- stored detect/active red/au/abort coverage CPU time
- --------- ------------ ------------ -------- --------
- Begin deterministic ATPG: #uncollapsed_faults=5472, abort_limit=250...
- Patn 1: #merges=320 #failed_merges=9 #faults=2977 #det=877 CPU=0.03 sec
- Patn 2: #merges=213 #failed_merges=12 #faults=2755 #det=323 CPU=0.07 sec
- Patn 3: #merges=315 #failed_merges=15 #faults=2438 #det=460 CPU=0.12 sec
- Patn 4: #merges=133 #failed_merges=18 #faults=2286 #det=226 CPU=0.15 sec
- Patn 5: #merges=225 #failed_merges=10 #faults=2060 #det=323 CPU=0.19 sec
- Patn 6: #merges=171 #failed_merges=9 #faults=1888 #det=255 CPU=0.22 sec
- Patn 7: #merges=190 #failed_merges=6 #faults=1697 #det=272 CPU=0.25 sec
- Patn 8: #merges=64 #failed_merges=11 #faults=1631 #det=92 CPU=0.27 sec
- Patn 9: #merges=106 #failed_merges=4 #faults=1515 #det=193 CPU=0.29 sec
- Patn 10: #merges=103 #failed_merges=5 #faults=1411 #det=150 CPU=0.31 sec

Result

- Test coverage
  - Uncollapsed Stuck Fault Summary Report
  - ----------
  - fault class code #faults
  - ---------- ---- ---------
  - Detected DT 5912
  - Possibly detected PT 0
  - Undetectable UD 101
  - ATPG untestable AU 1
  - Not detected ND 44
  - ----------
  - total faults 6058
  - test coverage 99.24%
  - ----------
  - Pattern Summary Report
  - ----------
  - #internal patterns 168
  - #basic_scan patterns 168
  - ----------

Fault Class

- Undectable
  - Cannot be tested by any means
- ATPG Untestable
  - Cannot be found using ATPG, but may be detected by other methods(functional tests)
- Not Detected
  - Cannot be found due to ATPG iterations limits or designs too complex



Reporting Faults

- Sets the parameters that control the fault manager
  - TEST > set_faults -summary verbose
- Set which kind of faults you want to see collapsed/uncollapsed
  - TEST > set_faults -report collapsed
  - TEST > report_summaries
- Display fault data
  - "-class" : Specifies a specific fault class to be reported
    - TEST > report_faults -class UD

Reporting Faults

- "-level [d] [m]" :

  Generates a fault report for specified hierarchical levels. The d argument specifies the hierarchical depth of the report and the m specifies a minimum number of faults required to display a given depth

  - TEST > report_faults -level 5 10

Writing Faults

- Writes fault data to external file
  - TEST > write_faults pre_norm_faults.rpt -all -replace
- Writes patterns to external file
  - TEST > write_patterns pre_norm_test_patterns.stil -format stil -replace
  - TEST > exit

Outline

- Introduction
- Design Compiler
- TetraMax
- Lab

Goal(1/2)

- Compare area , power, test coverage, ATPG runtime and # of test pattern differences for circuits with and without inserting scan-chain.
- For circuit without scan-chain, don't set any command related to scan in design compiler, includeing: compile -scan, preview_dft, insert_dft, set_scan_configuration, report_scan_path, create_test_protocol, write_test_protocol, write_scan_def
- For ATPG without scan-chain, please remember to add an ATPG constraint.
  - set_atpg -full_seq_time {600 86400}

Goal(2/2)

- For circuit without scan-chain running ATPG, use the following command: run_drc
- For circuit without scan-chain doing ATPG, use option -full_seq_atpg
  - TEST> set_atpg -full_seq_atpg

Result

pre_norm	Area	Power	Coverage (collapsed)	ATPG Run Time	Pattern
Non-Scanned (-full seq atpg)	100956	238uw	98.02%	211.50	539
Scanned	107248	240uw	99.63%	1.57	168

Homework

- Run pre_norm.v and s38584_seq.v
- Generate a result table like last slide.

Reference

- [1] SynopsysInc., “Design Compiler User Guide”, Dec. 2004.
- [2] SynopsysInc., “Design Compiler Command-Line Interface Guide”
- [3] SynopsysInc., “Design CompilerReference Manual”
- [4] IPCORE Lab Slide 2006, Tian-Sheuan Chang
- [5] VLSI Testing Course Slide, Jing-Jia Liou
- [6] CIC Training Center Slide, Hsin-Jung Huang