

Lab2

Scan Chain Insertion and ATPG Using Tessent

Prof: Chia-Tso Chao

TA: Yu-Teng Nien

2022-12-15

Outline

- Introduction
- Tessent Scan
- Tessent Fastscan
- Mixed Flow
- Lab

Outline

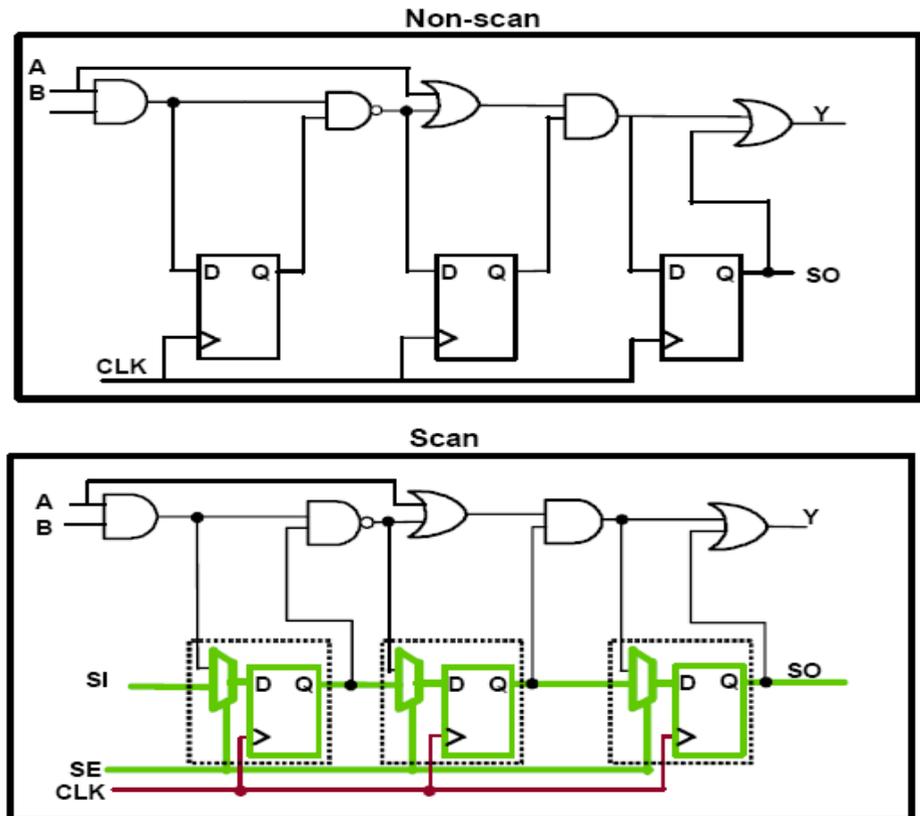
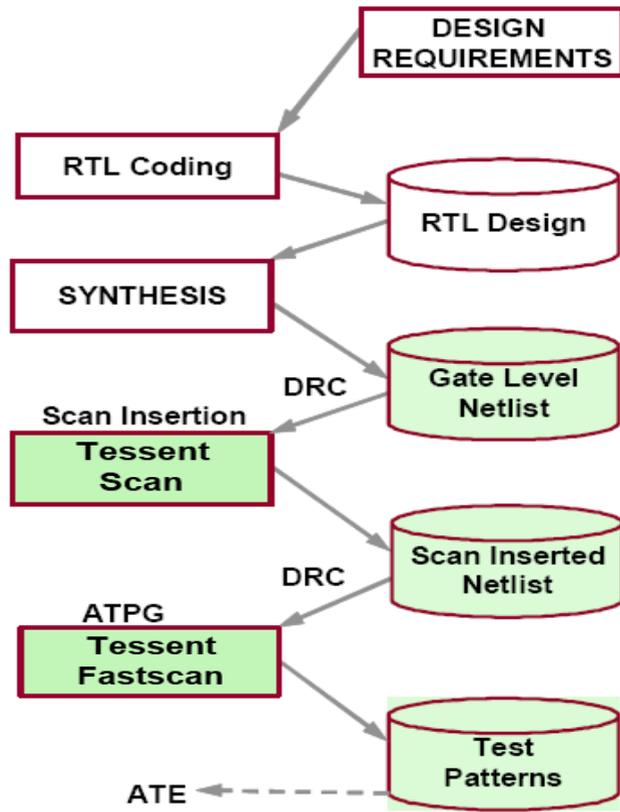
- Introduction
- Tessent Scan
- Tessent Fastscan
- Mixed Flow
- Lab

Introduction

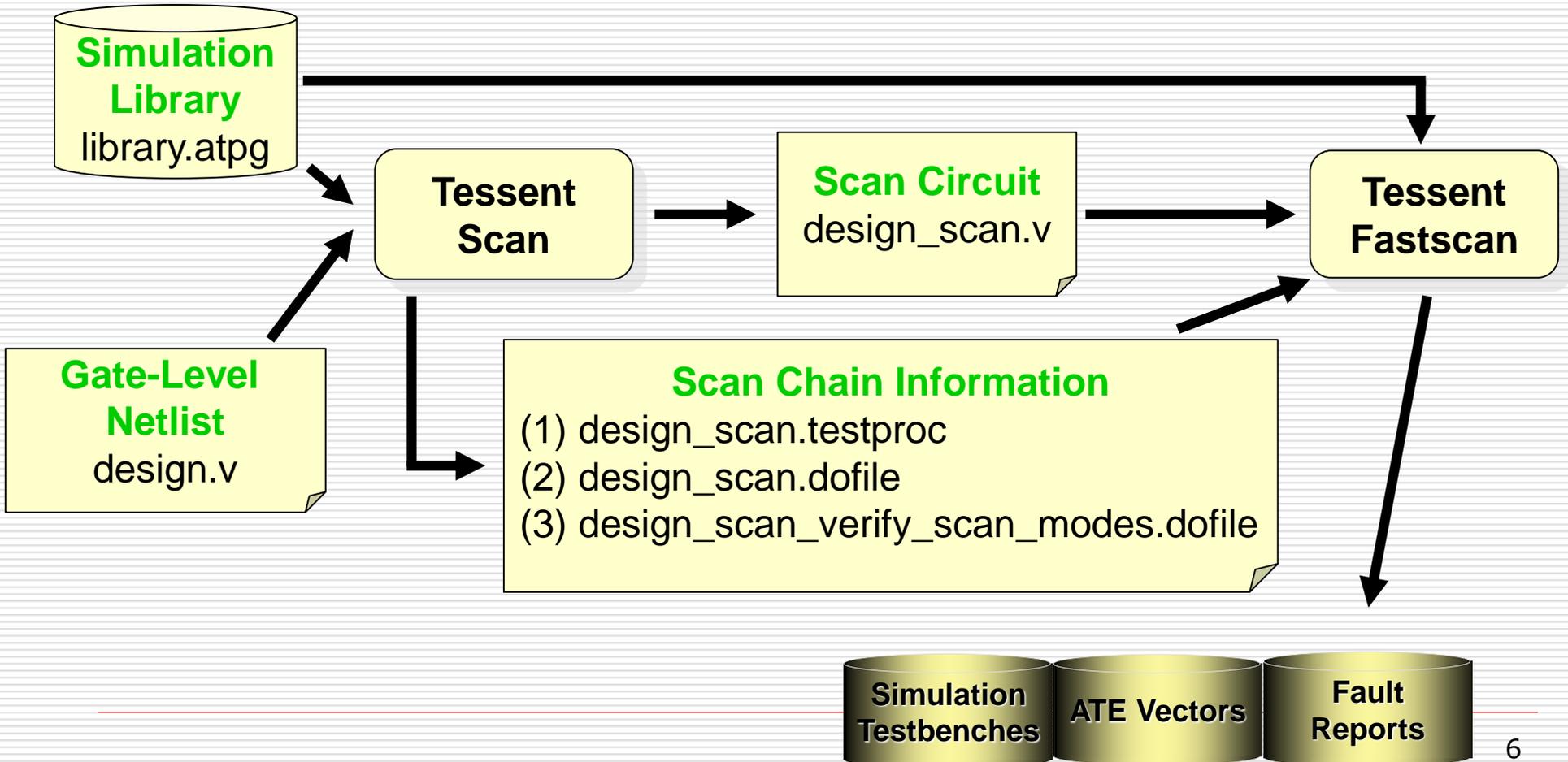
- This lab focuses on ATPG with tools from 2 different EDA vendors
 - Synopsys
 - Mentor Graphics
- Tessent Scan inserts scan chains
 - Basically replaces FFs with scan-FFs
- Tessent Fastscan performs ATPG and fault simulation

Insert Scan and ATPG Flow

Tool Flow



Input/Output Files



Outline

- Introduction
- Tessent Scan
- Tessent Fastscan
- Mixed Flow
- Lab

Invoke Tessent Scan

- Invoke Tessent shell
 - `$ tessent -shell`
- Default system mode is "SETUP"
 - You'll see the prompt shows
 - `SETUP> |`
- Set context to "dft -scan"
 - Which corresponds to Tessent Scan
 - `SETUP> set_context dft -scan`

Read Verilog and Library File

- Load cell library into the tool
 - Library is in Mentor Graphics' own format
 - `SETUP> read_cell_library I90sprvt.atpg`

- Read in Verilog source file
 - An already-synthesized gate-level netlist
 - `SETUP> read_verilog pre_norm_noscan.v`

Specify Top Module and Clock

- Specify the top level of the design
 - `SETUP> set_current_design pre_norm`
 - Clocks are primary input signals that synchronously change the state of sequential logic elements
 - `SETUP> add_clocks 0 clk` ← primary input to design
 - ↑
positive edge-triggered signal
 - Check the clock list
 - `SETUP> report_clocks`
-

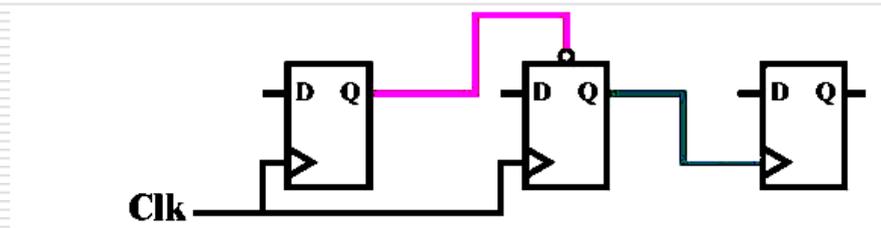
Setup Test Logic Configuration

- Test logic options make clock lines controllable to get a scannable design

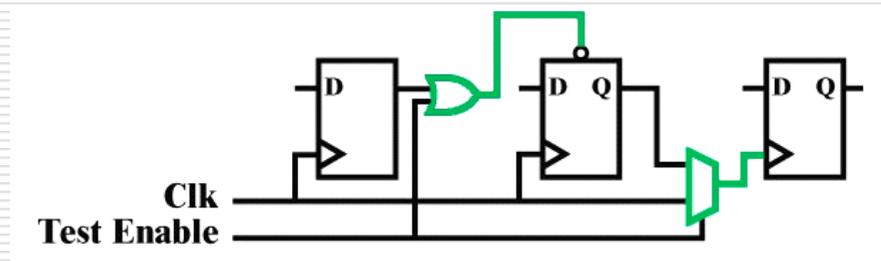
- `SETUP> set_test_logic`
-clock on -reset on

- Verify test logic configuration with

- `report_environment`



Non-scannable



Scannable after test logic insertion

Entering Analysis System Mode

- ❑ Enter netlist analysis mode and perform scan identification
 - `SETUP> set_system_mode analysis`
- ❑ Report detailed statistics report of scan identification
 - #Sequential instances
 - #Identified scan instances
 - #Scannable instances
 - `ANALYSIS> report_statistics`

Insert Scan Chain and View Report

- Specify number of scan chains
 - `ANALYSIS> set_scan_insertion_options -chain_count 10`
- Distributes scan cells over new scan chains
 - `ANALYSIS> analyze_scan_chains`
- Insert scan chain and add test logic circuitry
 - `ANALYSIS> insert_test_logic`
- Report all defined scan chains & test logic
 - `INSERTION> report_scan_chains`
 - `INSERTION> report_test_logic`

Output Scan Design for ATPG

□ Write out scan design and setup files

■ INSERTION> write_design

-output pre_norm_scan.v

-replace

■ INSERTION> write_atpg_setup pre_norm_scan

-replace

↓ <design>

<design>.testproc: test procedure file

<design>.dofile: setup information for ATPG

<design>_verify_scan_modes.dofile: TOP setup script for ATPG

Outline

- Introduction
- Tessent Scan
- Tessent Fastscan
- Mixed Flow
- Lab

Invoke Tessent Fastscan and ATPG Setup

- Change system mode to setup
 - `INSERTION> set_system_mode setup`
- Setup ATPG and switch to Tessent Fastscan
 - `SETUP> dofile`
`pre_norm_scan_verify_scan_modes.dofile`
- Select fault type: stuck-at, IDDDQ, transition, path delay, bridge, etc.
 - `ANALYSIS> 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
 - `ANALYSIS> create_patterns -auto`

Generate Patterns (2/2)

- Without the "auto" option, you can specify your own configurations using these commands
 - ANALYSIS> set_atpg_limits
 - cpu_seconds [integer]
 - test_coverage [real]
 - pattern_count [integer]
 - ANALYSIS> set_abort_limit [integer]
 - ANALYSIS> create_patterns

During ATPG Run

- ATPG is performed one pass after another

```
Simulation performed for #gates = 3420 #faults = 5542
system mode = ATPG    pattern source = internal patterns
```

```
-----
#patterns  test      #faults  #faults  # eff.   # test   process   RE/AU/AAB
simulated  coverage  in list  detected  patterns patterns CPU time
deterministic ATPG invoked with comb/seq abort limit = 300/100
---
64         87.16%    781     4761     60      60      1.15 sec  0/0/15
---
128        95.93%    243     450      51      111     1.23 sec  87/1/91
---
192        98.81%    70      143      50      161     2.33 sec  117/1/103
---
229        99.73%    15      55       31      192     2.58 sec  117/1/103
---
2.59 sec
```

ATPG Result

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

Statistics Report Stuck-at Faults	
Fault Classes	#faults (total)
FU (full)	6098
U0 (unobserved)	15 (0.25%)
DS (det_simulation)	5409 (88.70%)
DI (det_implication)	540 (8.86%)
UU (unused)	16 (0.26%)
RE (redundant)	117 (1.92%)
AU (atpg_untestable)	1 (0.02%)
Coverage	
test_coverage	99.73%
fault_coverage	97.56%
atpg_effectiveness	99.75%
#test_patterns	192
#simulated_patterns	229
CPU_time (secs)	300.5

View Fault Report

- Display fault information
 - `ANALYSIS> report_faults -all`
- A fault is determined by (i) a fault value and (ii) a fault site
- Each fault is associated with a fault class/code

Fault value:
Either 0 (for stuck-at-0)
or 1 (for stuck-at-1)

Fault code

Fault site

```
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/i1
0  AU  /IS20/en
1  AU  /IS20/en
0  AU  /IS2/en
1  AU  /IS2/en
```

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 failure, 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 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 whose paths all blocked by tied logic
- TI: Tied
 - Point of the fault value is always same (e.g. SA 0 at AND2 with complementary inputs)
- RE: Redundant
 - Faults undetectable after exhausting all patterns and need dedicated analysis to verify redundancy
 - `ANALYSIS> identify_redundant_faults`

Test Coverage Formula Comparison

□ TetraMAX

$$\text{test_coverage} = \frac{DT + (PT * \text{posdet_credit})}{\text{all_faults} - (UD + AU * \text{au_credit})}$$

Annotations for TetraMAX formula:
- *possible detected* points to *PT*
- *default 50%* points to *posdet_credit*
- *default 0* points to *au_credit*

□ Tessent Fastscan

$$\text{test_coverage} = \frac{DT + (PD * \text{posdet_credit})}{\text{testable}} * 100$$

Annotation for Tessent Fastscan formula:
- *default 50%* points to *posdet_credit*

$$\text{fault_coverage} = \frac{DT + (PD * \text{posdet_credit})}{\text{full}} * 100$$

$$\text{ATPG_effectiveness} = \frac{DT + UT + AU + PU + (PT * \text{posdet_credit})}{\text{full}} * 100$$

Save Patterns

- Save patterns that are generated via ATPG
- Various formats including binwgl, ctl2005, stil2005, stil999, Verilog, VHDL, wgl, zycad, tstl2, utic
 - ANALYSIS> write_patterns pre_norm_scan.pat
-verilog -proc -replace
 - ANALYSIS> write_patterns pre_norm_scan_tstl2.pat
-TSTL2 -replace
 - ANALYSIS> exit

↑
Toshiba Standard Tester Interface
Language 2

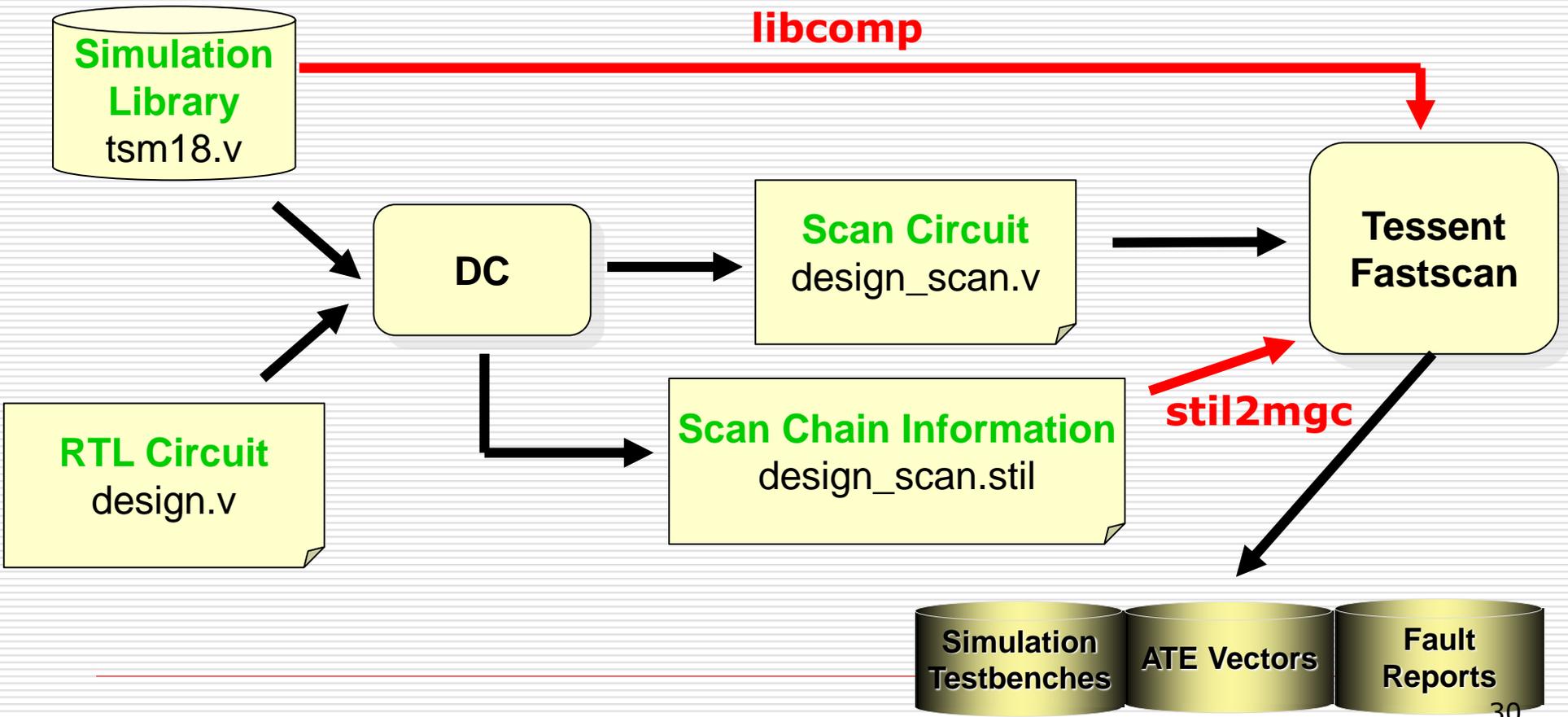
Outline

- Introduction
- Tessent Scan
- Tessent Fastscan
- **Mixed Flow**
- Lab

Mixed Flow

- ❑ Synopsys Design Compiler is way better at mapping design 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
 - Tessent Fastscan to perform ATPG

Input/Output Files



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 and .proc
- Scan design
 - Gate-level netlist with scan chain inserted

Convert STIL File

- Use “`stil2mgc`” to convert STIL file from Design Compiler into Tessent-compatible dofile and test procedure file
 - `$ stil2mgc pre_norm_scan.stil`
 - It generates both files for setup in Fastscan
 - `pre_norm_scan.stil.do`
 - `pre_norm_scan.stil.proc`

Perform ATPG using Tessent

- ❑ Read scan circuit from Design Compiler to perform ATPG
 - `$ tessent -shell`
 - `SETUP> set_context patterns -scan`
 - `SETUP> read_cell_library I90sprvt.atpg`
 - `SETUP> read_verilog pre_norm_scan.v`
 - `SETUP> dofile pre_norm_scan.stil.do`
 - `SETUP> set_system_mode analysis`
 - `ANALYSIS> set_fault_type stuck`
 - `ANALYSIS> create_patterns -auto`
 - `ANALYSIS> report_statistics`
-

Outline

- Introduction
- Tessent Scan
- Tessent Fastscan
- Mixed Flow
- Lab

Lab Objective

- Compare the following during ATPG using the DC+TMAX, DC+TS and TS flows
 - Total fault number
 - Test coverage
 - Pattern count
 - Run time (s)
- Run on circuit “pre_norm”
 - DC+TMAX, DC+TS: “pre_norm.v” in ~/lab1/
 - TS: “pre_norm_noscan.v” in ~/lab2/

Example of Lab Result

Flow	#Faults	Test Coverage	#Patterns	Run time
DC+TMAX	71298	100%	138	0.83s
TS	122072	100%	201	0.66s
DC+TS	75208	100%	203	0.51s

References

□ Mentor Graphics (Siemens)

- Tessent Scan and ATPG User's Manual, v2019.3

□ Synopsys

- TetraMAX ATPG User Guide, J-2014.09-SP1
- TestMAX ATPG User Guide, P-2019.03