

Project 5 Tutorial: Benchmark Circuit s27

18-765: Digital Systems Testing and Testable Design

In this tutorial the general procedure of Project 5 will be demonstrated through the use of a much smaller sequential circuit. ISCAS Benchmark circuit s27 has four inputs, one output, eight logic gates, and three D flip flops. All required files for this process can be found in *example_s27/* subdirectory of the Project 5 directory.

For any questions or further information about the tasks described in this tutorial please reference *Project 5 Scripts Help*, or the Siemens Tessent manuals.

Table of Contents

Tool Setup

[Background](#)

[Steps](#)

ATPG with Combinational Circuit

[Background](#)

[Steps](#)

Test Vector Conversion and Serialization

[Background](#)

[Steps](#)

DFT Circuit Simulation

[Background](#)

[Steps](#)

Tool Setup

Background

As discussed in the project document, there are two primary tools that will be used in this project: Siemens Tessent for ATPG, and Siemens KaleidoScope for Fault Simulation. To access these tools, you will need to have certain environment variables configured in your shell.

Steps

1. Copy the Project 5 directory to your own working area.
2. Navigate to the *siemens*/subdirectory, which contains two files.
3. Source the appropriate file to set environment variables:
 - a. `source siemens.csh` for `csh`
 - b. `source siemens.bash` for `bash`
4. Confirm that you can properly launch the tools
 - a. For Tessent, try the below commands and confirm similar output:

```
$ which tessent
/afs/ece.cmu.edu/support/mgc/mgc.release/tessent_2023_3/bin/tessent
$ tessent -shell
// Tessent Shell 2023.3 Tue Aug 15 20:24:46 GMT 2023
// Unpublished work. Copyright 2023 Siemens
//
// This material contains trade secrets or otherwise confidential
// information owned by Siemens Industry Software Inc. or its affiliates
// (collectively, "SISW"), or its licensors. Access to and use of this
// information is strictly limited as set forth in the Customer's
// applicable agreements with SISW.
//
// Siemens software executing under x86-64 Linux on Mon Nov 13 11:39:09 EST 2023.
// 64 bit version
// Host: <hostname> (322197 MB RAM, 8191 MB Swap)
//
SETUP>
```

- b. For KaleidoScope, try the below commands:

```
$ which KaleidoScope
/afs/ece.cmu.edu/support/mgc/mgc.release/austemper_2023_3/linux_x86_64/bin/KaleidoScope
$ KaleidoScope --version
#
# Version 2024.1-internal CL-5708550 linux_x86_64 9 Nov 2023
```

ATPG with Combinational Circuit

Background

In order to simulate the final design of your DFT circuit you must create a series of test vectors. The first step in this process is to convert the original sequential circuit to the combinational circuit. You will create a second Verilog module that will contain all of the combinational logic from the sequential circuit. All of the inputs to D flip flops will become primary outputs of the combinational circuit, and the flip flop outputs will become primary inputs. The D flip flops and any of their other signals (clocks, reset) are removed from the module.

Steps

1. Navigate to the *example_s27*/subdirectory of your Project 5 directory. This contains the files needed for constructing and running ATPG on the sample design.
2. Open *verilog/s27.v* and *verilog/s27_comb.v* alongside one another. Compare the two and note the differences described above. When you perform this step in your own application of Project 5, the combinational version of your circuit should exhibit these same changes.
3. Now it's time to generate tests for *s27_comb.v*. Navigate into the *tessent*/subdirectory and open the file *s27_comb_atpg.tcl*. Note the commands and their meaning, as you will likely need to replicate these steps with your design.

```
≡ s27_comb_atpg.tcl x
18765 > ece765 > projects > project5 > example_s27 > tessent > ≡ s27_comb_atpg.tcl
1  set_context pattern -scan
2
3  # read the verilog file and define the top-level module
4  read_verilog ../verilog/s27_comb.v
5  set_current_design s27_comb
6
7  # put Tessent into pattern generation mode
8  set_system_mode analysis
9
10 # add all stuck-at faults to the design
11 add_faults -all
12
13 # generate patterns
14 create_patterns
15
16 # write the patterns to a file with ASCII format
17 write_patterns s27_comb_tests.ascii
18
19 # write the faults to a file
20 report_faults > s27_comb.faults
21
22 exit -d
```

4. Now run the command `tessent -shell -dofile s27_comb_atpg.tcl`. Towards the bottom of the terminal output you should see a report like the one below and should have a fault coverage of 100%.

Statistics Report Stuck-at Faults	
Fault Classes	#faults (total)
FU (full)	78
DS (det_simulation)	78 (100.00%)
Coverage	
test_coverage	100.00%
fault_coverage	100.00%
atpg_effectiveness	100.00%
#test_patterns	8
#simulated_patterns	11
CPU_time (secs)	7.4

The file `s27_comb_tests.ascii` contains the tests generated targeting the faults in the `s27_comb` module.

Test Vector Conversion and Serialization

Background

After the test vectors have been created for the combinational version of your circuit, the bits from those vectors will need to be manipulated in order to achieve the correct testing conditions for the DFT version of the circuit. For this tutorial, the DFT circuit implements internal scan. When working with a scan-based design, the vectors generated for the combinational circuit will need to be serialized to apply multiple bit values to a single input.

This portion of the tutorial contains three main steps involving the scripts accompanying this project. The first converts the output from Tessent to a file that is more easy to read and work with. The second step will serialize those test vectors. The serialized vectors are then converted back to a file format readable by KaleidoScope.

Steps

1. From the *example_s27/* directory, run the below command

```
python ../scripts/ascii_to_etr.py tessent/s27_comb_tests.ascii s27_comb.etr
```

A new file will be created. *s27_comb.etr* contains the test vectors from the Tessent-created ASCII file. You may wish to view these test vector files alongside one another to observe the differences in the two file types and the advantages of each.

2. Now take a look at the mapping file *s27_comb.p5map*. This file maps the combinational signals to the scan signals. You will need to create one of these files for *s9234.v*.
3. From the *example_s27* directory, serialize the test vectors in *s27_comb.etr* according to the mapping file *s27_comb.p5map* using the following command.

```
python ../scripts/p5serialize.py s27_comb.etr s27_comb.p5map s27_scan.Setr
```

This will create a new file of serialized test vectors named *s27_scan.Setr*. Although the vectors have been serialized in this file there are still modifications that need to be made in order to apply these test vectors to the new DFT version of the circuit.

4. Open *verilog/s27_scan.v* alongside the modified file of test vectors *s27_scan.Setr*.
Within *s27_scan.Setr*, change the names of the SCAN_IN/SCAN_OUT pins to

match the names in the *verilog/s27_scan.v* circuit. In this situation, these are the only changes that need to be made.

In creating the test vectors for your own Project 5 DFT circuit, you may need to add in test vectors to initialize the circuit. This may include a reset for flip-flops, or the use of other control signals to prepare for scan. Ensure that these vectors are correctly structured for your circuit.

The bits from your *s27_comb* test patterns have been placed in a format that allows them to be correctly applied to the DFT circuit *verilog/s27_scan.v*. Viewing this circuit may help in your understanding of how the bits are placed. You will need to perform the same type of bit manipulation when creating the test vectors to be used on your Project 5 DFT circuit.

5. From your *example_s27/* directory, convert the test vectors in *s27_scan.etr* to the VCD format using the following command.

```
python ../scripts/etr_to_vcd.py s27_scan.Setr s27_scan.vcd s27_scan
```

A new VCD file will be created. You may find it useful to open *s27_scan.vcd* alongside the *s27_scan.Setr* to compare the file types.

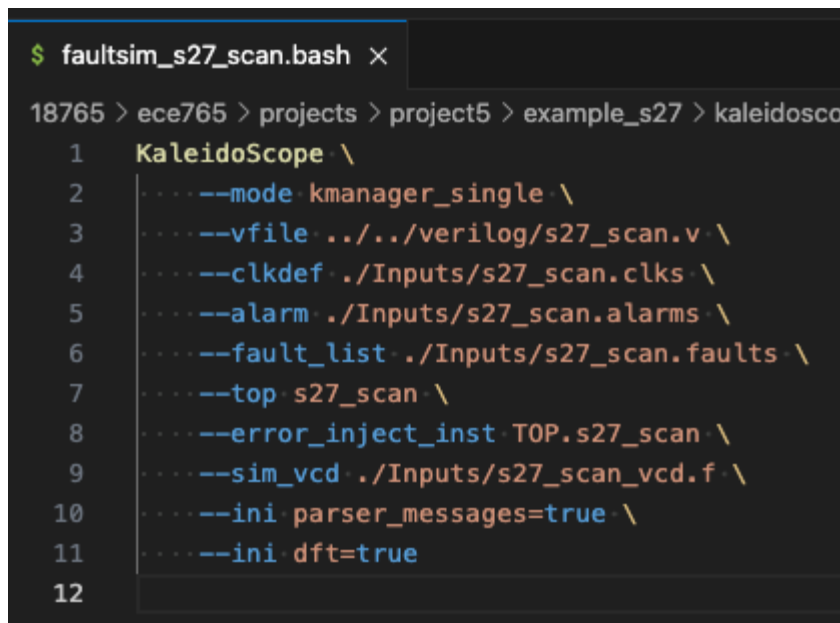
DFT Circuit Simulation

Background

After you have created your DFT circuit for Project 5, you will need to simulate it to find how well it detects faults. You will use the test vectors that you have prepared in the previous steps to perform this simulation. After the simulation is performed a fault coverage percentage will be printed, essentially grading how well the combination of your circuit and test vectors performed. The ultimate goal of Project 5 is to make this percentage as high as you can.

Steps

1. Navigate to the *example_s27/kaleidoscope/* subdirectory.
2. We'll simulate your newly-created test patterns using the *faultsim_s27_scan.bash* script. Open the file and examine the options used in the command.



```
$ faultsim_s27_scan.bash x
18765 > ece765 > projects > project5 > example_s27 > kaleidoscope
1  KaleidoScope \
2  |   --mode kmanager_single \
3  |   --vfile ../../verilog/s27_scan.v \
4  |   --clkdef ./Inputs/s27_scan.clks \
5  |   --alarm ./Inputs/s27_scan.alarms \
6  |   --fault_list ./Inputs/s27_scan.faults \
7  |   --top s27_scan \
8  |   --error_inject_inst TOP.s27_scan \
9  |   --sim_vcd ./Inputs/s27_scan_vcd.f \
10 |   --ini_parser_messages=true \
11 |   --ini_dft=true
12
```

There are a number files used in this command, present in the *Inputs/* subdirectory:

- *../../verilog/s27_scan.v* is the DFT-inserted scan circuit
- *./Inputs/s27_scan.clks* defines the clock signals for the circuit
- *./Inputs/s27_scan.alarms* defines circuit outputs where faults may be observed
- *./Inputs/s27_scan.faults* defines a target list of stuck-at faults to be simulated
 - Note that this file uses pin names from the DFT-inserted circuit, not the original functional circuit.
- *./Inputs/s27_scan_vcd.f* lists the paths to the VCD that will be simulated

Please review each of these files, as you will need to create them for your own design.

In this stage of the design, **you are not generating test patterns but simulating them.**

- Finally, run the script with command `./faultsim_s27_scan.bash`. Note the final table reporting the classification of detection status for the faults in the test set. In this case, we add the “Detected” coverage of 86.05% to the “Potential Detected” coverage of 6.98% to achieve the final fault coverage of 93.03%.

```
=====
DFT Faults Summary
=====
```

Total Number of Faults	:	86
Detected	:	74 (86.05%)
Potential Detected	:	12 (6.98%, PT credit = 0.5; default)
Not Detected	:	0 (0.00%)
** Controlled Observed Internal	:	0
** Reached BlackBox	:	0
** Missing Simulation Data	:	0
** Controlled Unobserved	:	0
** Uncontrolled Unobserved	:	0
** Constant Fault	:	0
** Unused Fault	:	0

```
=====
```

- If you haven’t achieved your target fault coverage, you’ll likely want to debug the faults that were left undetected or potentially detected. You can review the `Outputs/s27_scan.DFT.rpt` file for detection status of individual faults.

```
s27_scan.DFT.rpt x
```

18765 > ece765 > projects > project5 > example_s27 > kaleidoscope > Outputs > s27_scan.DFT.rpt

	#FaultType	FaultClass	FaultNode
1	0	PT	s27_scan.CK
2	1	PT	s27_scan.CK
3	0	DS	s27_scan.G0
4	1	DS	s27_scan.G0
5	0	DS	s27_scan.G1
6	1	DS	s27_scan.G1
7	0	DS	s27_scan.G2
8	1	DS	s27_scan.G2
9	0	DS	s27_scan.G3
10	1	DS	s27_scan.G3
11	0	DS	s27_scan.G17
12	1	DS	s27_scan.G17
13	0	DS	s27_scan.SFF_0.D
14	1	DS	s27_scan.SFF_0.D
15	0	PT	s27_scan.SFF_0.CK
16	1	PT	s27_scan.SFF_0.CK
17	0	PT	s27_scan.SFF_0.Q
18	1	DS	s27_scan.SFF_0.Q
19	0	DS	s27_scan.SFF_1.D
20	1	DS	s27_scan.SFF_1.D
21	0	PT	s27_scan.SFF_1.CK
22	1	PT	s27_scan.SFF_1.CK
23	0	DS	s27_scan.SFF_1.Q
24	1	DS	s27_scan.SFF_1.Q
25	0	DS	s27_scan.SFF_2.D
26	1	DS	s27_scan.SFF_2.D
27	0	PT	s27_scan.SFF_2.CK
28	1	PT	s27_scan.SFF_2.CK
29	0	DS	s27_scan.SFF_2.Q
30	1	DS	s27_scan.SFF_2.Q
31	0	DS	s27_scan.NOT_0.i
32	1	DS	s27_scan.NOT_0.i