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 *p5example_s27.zip*.

For any questions or further information about the tasks described in this tutorial please reference *Project 5 Cadence Encounter Test Help*, *Project 5 Scripts Help*, or the help documents within Cadence Encounter Test.

Table of Contents

Tutorial Setup

Background

If this is your first time using Cadence Encounter Test, please visit the *Project 5 Cadence Encounter Test Help* document and read its introduction. This will describe the required setup for Encounter Test and how to start its GUI.

The documentation for this Project 5 will be accompanied by two *.zip* files. The first, *p5userscripts.zip* contains different scripts that will help you in working with the test vectors of this file. The script *p5convert.py* (using modules from *p5fileconvert.py*) will help you in converting between test vector formats. *p5serialize.py* is used to to serialize the test vectors in order to apply them to a newly designed, scan-based DFT circuit.
The second *.zip* file, entitled *p5example_s27.zip* contains the files used in this tutorial. The purpose of each of these files will become more clear as you perform the tasks in this tutorial.

Steps

Create a directory that will hold your files for Project 5. This directory will be referenced as `project5` in this tutorial.
Within that directory create a directory which will hold the working directories for the Cadence Encounter Test projects you will use in completing Project 5. Throughout the tutorial this directory will be referenced as `project5/cet_workdirs/`.

Unzip both of the accompanying files, creating two directories with names that match the *.zip* files.

```
      unzip p5example_s27.zip
      unzip p5userscripts.zip
```

Move these newly-made directories into the parent directory `project5/`.

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.

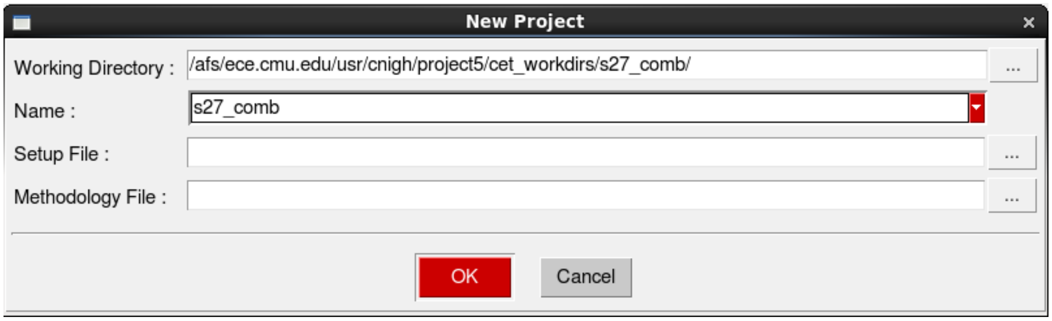
Open *s27.v* and *s27_comb.v* alongside one another. Compare the two and note the differences described above. When you perform this step in your own application of Project 5, the combinational version of your circuit should exhibit these same changes.

Steps

Open the Cadence Encounter Test GUI

Select **File □ New …**

Name your project and its working directory "s27_comb"
Click ***OK***



New Project

Working Directory : /afs/ece.cmu.edu/usr/cnigh/project5/cet_workdirs/s27_comb/

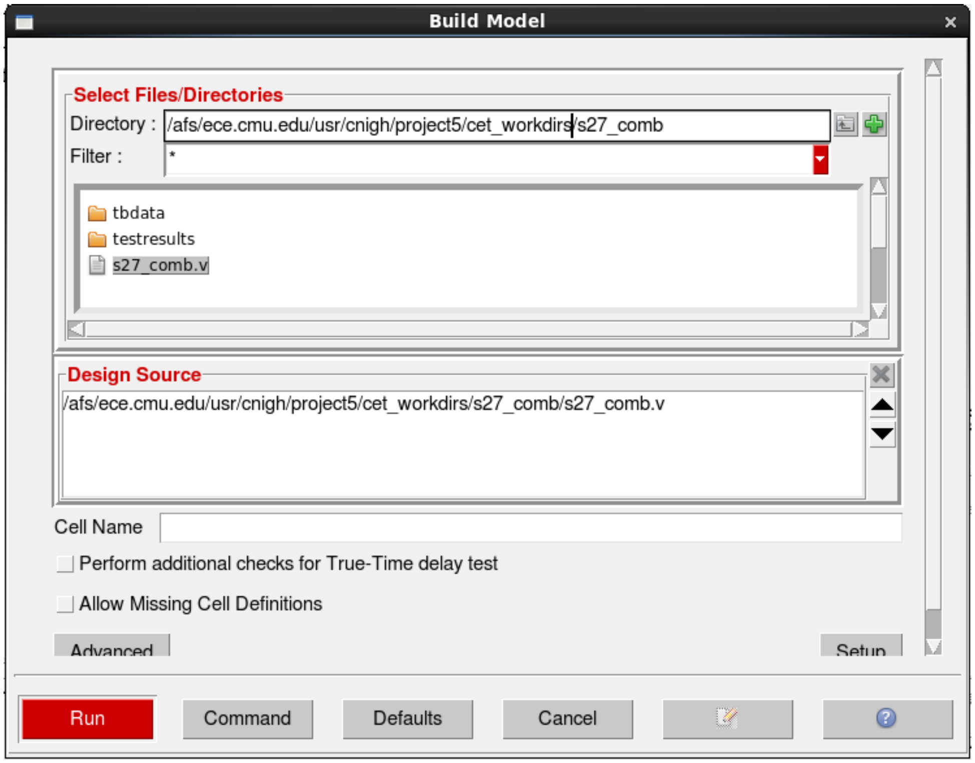
Name : s27_comb

Setup File :

Methodology File :

In a terminal, move (or copy) the file *s27_comb.v* from `project5/p5example_s27/` to the project's working directory `project5/cet_workdirs/s27_comb/`.

In Encounter Test, select **Verification □ Build Models □ Model …**
Navigate to the working directory and select “s27_comb.v” from the list of files.
Click ***Run***



Directory : /afs/ece.cmu.edu/usr/cnigh/project5/cet_workdirs/s27_comb

Filter : *

- tbdata
- testresults
- s27_comb.v

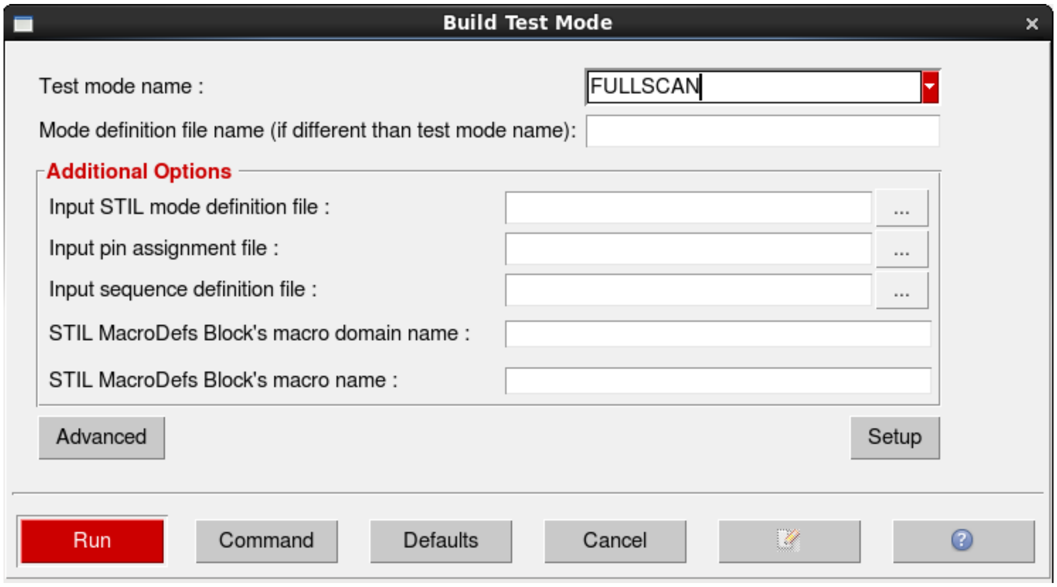
/afs/ece.cmu.edu/usr/cnigh/project5/cet_workdirs/s27_comb/s27_comb.v

Cell Name

□ Perform additional checks for True-Time delay test

□ Allow Missing Cell Definitions

Select **Verification □ Build Models □ Test Mode …**
Type “FULLSCAN” in for *Test mode name*
Click ***Run***



Test mode name : FULLSCAN

Mode definition file name (if different than test mode name):

Input STIL mode definition file :

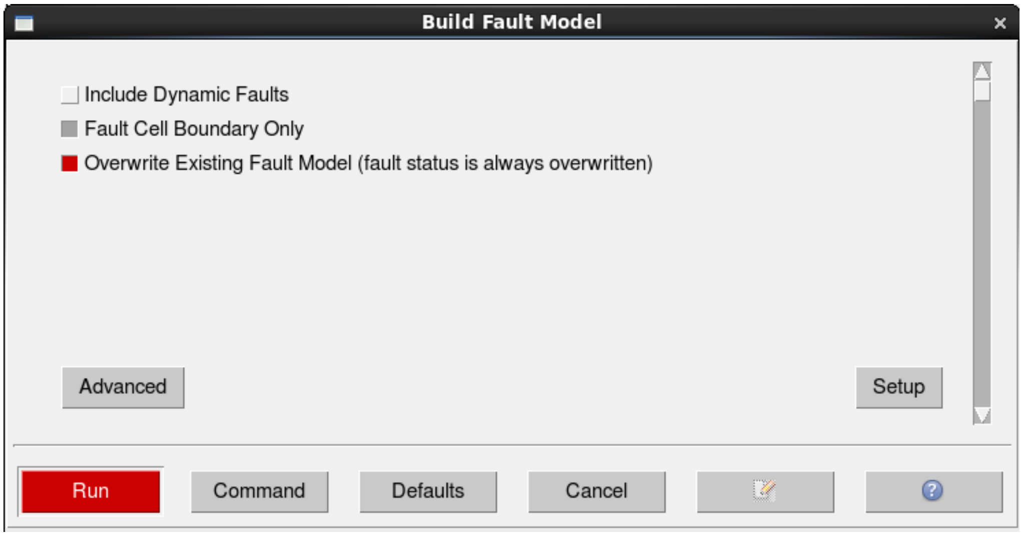
Input pin assignment file :

Input sequence definition file :

STIL MacroDefs Block's macro domain name :

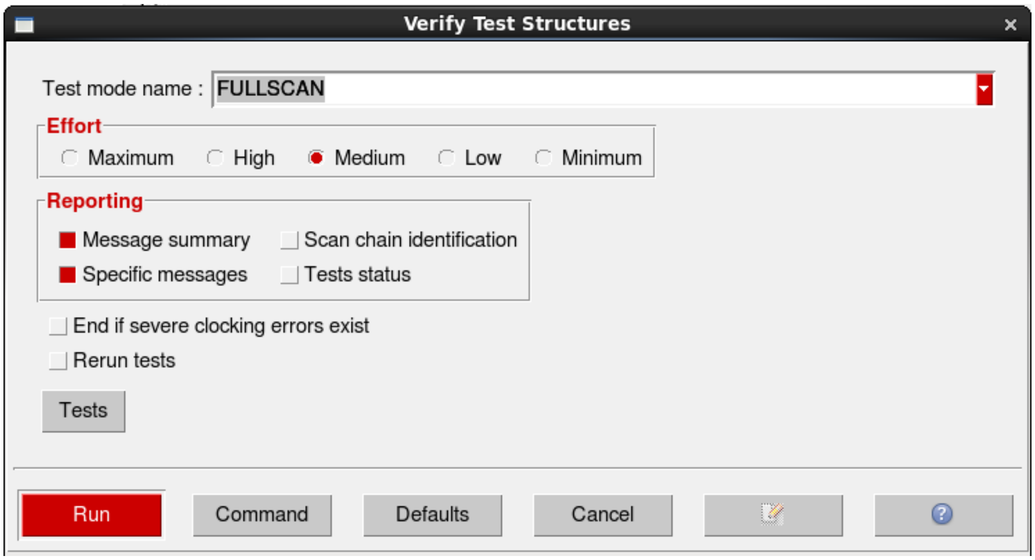
STIL MacroDefs Block's macro name :

Select **Verification □ Build Models □ Build Fault Model …**
Deselect *Include Dynamic Faults*
Click ***Run***


Build Fault Model

□ Include Dynamic Faults
■ Fault Cell Boundary Only
☒ Overwrite Existing Fault Model (fault status is always overwritten)

Select **Verification □ Verify □ Test Structures …**
Select "FULLSCAN" for the *Test mode name*
Click ***Run***


Verify Test Structures

Test mode name : FULLSCAN

Effort
○ Maximum ○ High ● Medium ○ Low ○ Minimum

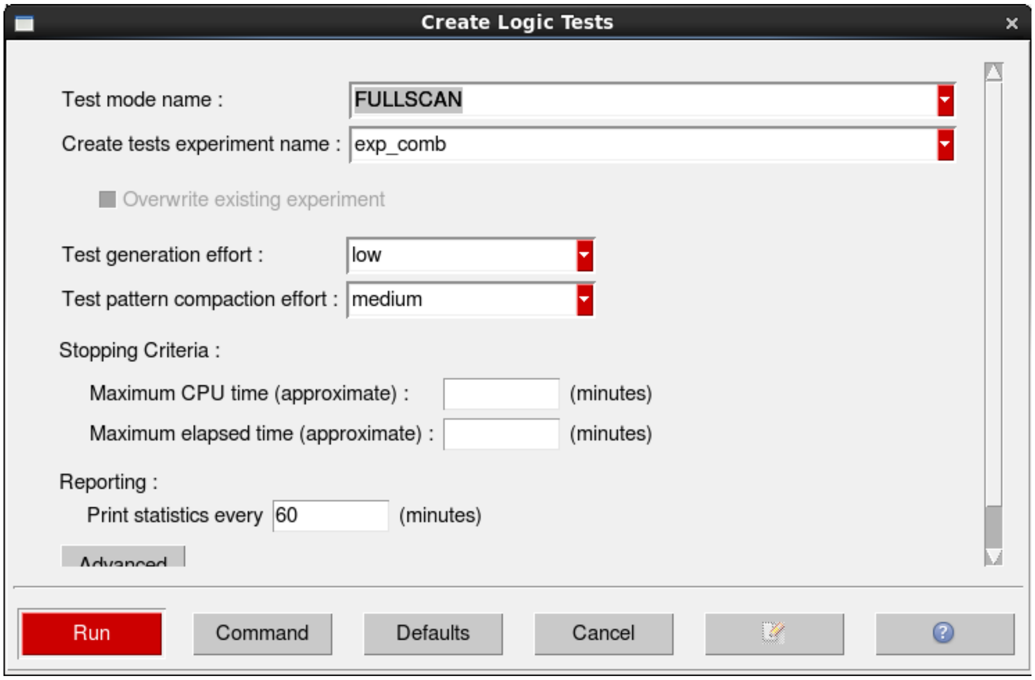
Reporting
☒ Message summary □ Scan chain identification
☒ Specific messages □ Tests status

□ End if severe clocking errors exist
□ Rerun tests

Select **ATPG □ Create Tests □ Specific Static Tests □ Logic …**
Select “FULLSCAN” for the *Test mode name*

Type “exp_comb” for *Create tests experiment name*
Click ***Run***

Confirm that 100% fault coverage is reached in the Encounter Test GUI’s session log.



Create Logic Tests

Test mode name : FULLSCAN

Create tests experiment name : exp_comb

□ Overwrite existing experiment

Test generation effort : low

Test pattern compaction effort : medium

Stopping Criteria :

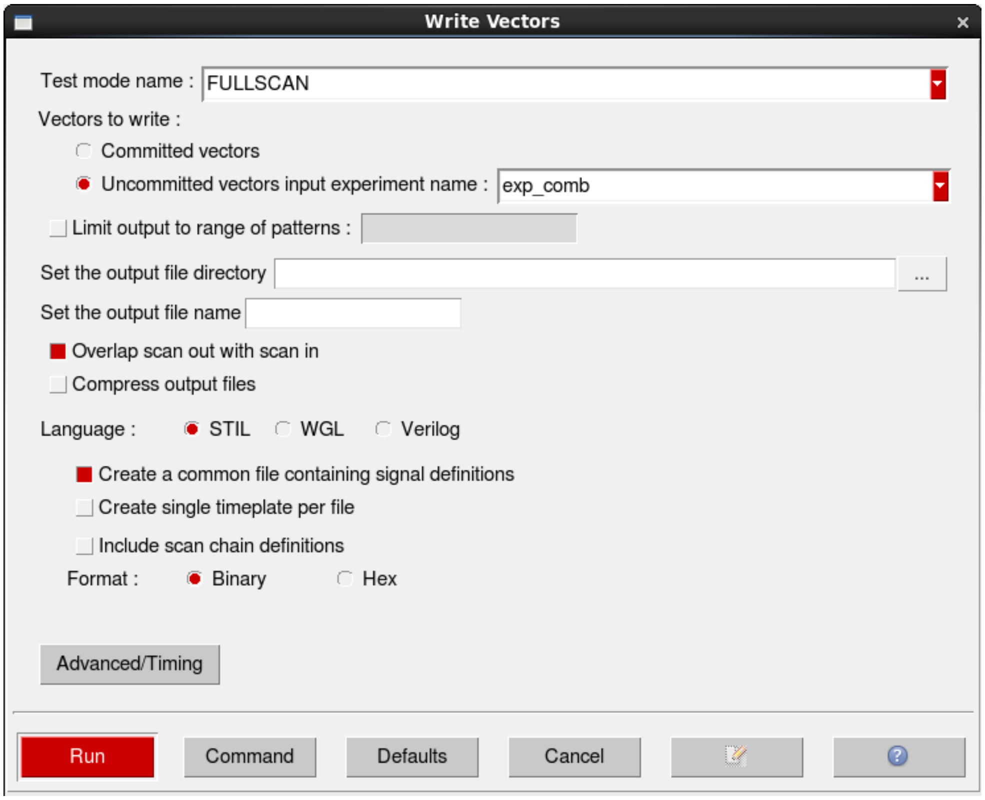
Maximum CPU time (approximate) : (minutes)

Maximum elapsed time (approximate) : (minutes)

Reporting :

Print statistics every 60 (minutes)

Select **ATPG □ Write Vectors …**
Select “FULLSCAN” for the *Test mode name*
For *Vectors to write*, select *Uncommitted vectors* and “exp_comb”
For *Language*, select “STIL”
Click ***Run***



Test mode name : FULLSCAN

Vectors to write :

○ Committed vectors
● Uncommitted vectors input experiment name : exp_comb

□ Limit output to range of patterns :

Set the output file directory

Set the output file name

■ Overlap scan out with scan in
□ Compress output files

Language : ● STIL ○ WGL ○ Verilog

■ Create a common file containing signal definitions
□ Create single timeplate per file
□ Include scan chain definitions

Format : ● Binary ○ Hex

New STIL files containing signal definitions and the generated test vectors will be created in the directory `s27_comb/testresults/stil/`.

Close Encounter Test

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 Encounter Test 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 Encounter Test.

Steps

In a terminal, navigate to the `project5/p5example_s27/` directory and create a new directory entitled `s27vec_temp`. From this new directory, run the *p5convert.py* script using the following command.

```
../../p5userscripts/p5convert.py
    ../../cet_workdirs/s27_comb/testresults/stil/STIL.FULLSCAN.exp_comb.logic.ex1.ts1
    .stil -f stil -t etr --map s27_comb
```

Two new files will be created in your `s27vec_temp` directory. The first, *s27_comb.etr* contains the test vectors from the Encounter Test-created STIL 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.

The other file in your `s27vec_temp` directory will be named *s27_comb.Tp5map* and is a template for the mapping file used in the *p5serialize.py* script. Open this template and view it alongside the finalized version of the mapping file *s27_comb.p5map* in the directory `/project5/p5example_s27/`. Take note of the additions, as you will need to perform similar actions when creating the mapping file for your own Project 5 design.

From your `s27vec_temp` directory, serialize the test vectors in *s27_comb.etr* according to the mapping file *s27_comb.p5map* using the following command.

```
../../p5userscripts/p5serialize.py s27_comb.etr ../s27_comb.p5map s27_scan
```

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. Open this file alongside the modified file of test vectors *s27_scan.etr* in the directory `/project5/p5example_s27/`. Note that the names of the scan in and scan out pins have changed to match the names in the *s27_scan.v* circuit. In this situation, these are the only changes that have been 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 have been placed in a manner which will allow them to be correctly applied to the DFT circuit *s27_scan.v* in the directory `/project5/p5example_s27/`. 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.

From your `s27vec_temp` directory, convert the test vectors in *s27_scan.etr* to the STIL format using the following command.

```
../../p5userscripts/p5convert.py ../s27_scan.etr -f etr -t stil s27_scan
```

Two new STIL files will be created. You may find it useful to open the file *s27_scan.vectors.stil* alongside the *s27_scan.etr* to compare the file types.

DFT Circuit Simulation

Background

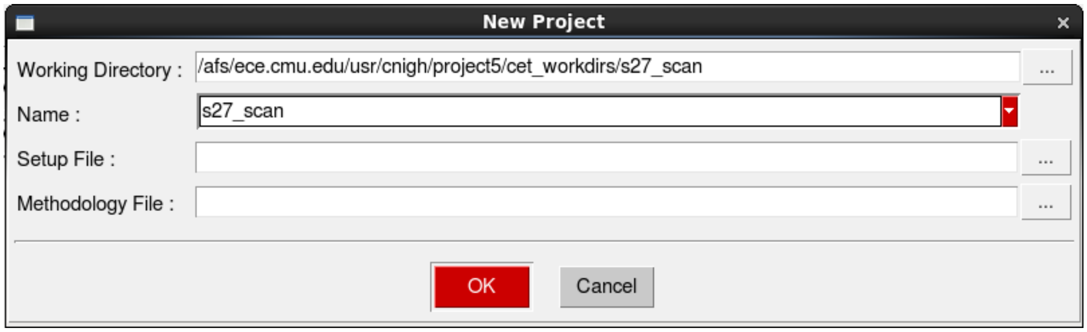
After you have created your DFT circuit for Project 5, you will need to simulate it to find how well it detects faults. You will use the test vectors that you have prepared in the previous steps to perform this simulation. After the simulation is performed a fault coverage percentage will be printed, 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

Open the Cadence Encounter Test GUI

Select **File □ New …**

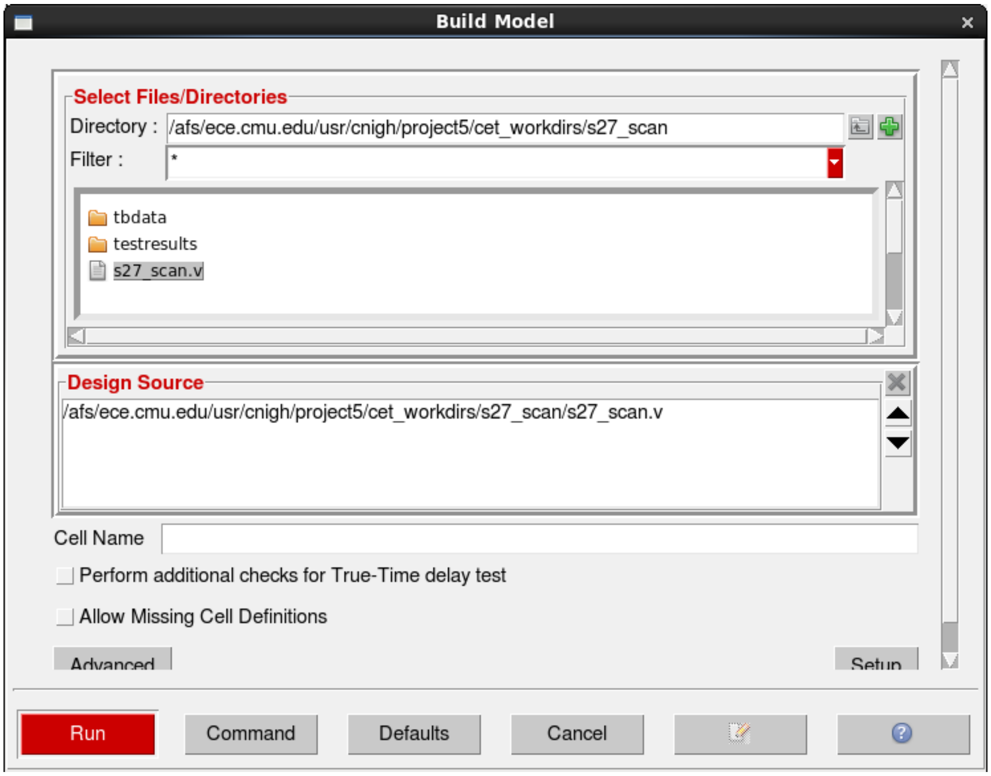
Name your project and its working directory “s27_scan”
Click ***OK***



Working Directory : /afs/ece.cmu.edu/usr/cnigh/project5/cet_workdirs/s27_scan
Name : s27_scan
Setup File :
Methodology File :

In a terminal, move (or copy) the file *s27_scan.v* from `project5/p5example_s27/` to the project’s working directory `project5/cet_workdirs/s27_scan/`.

In Encounter Test, select **Verification □ Build Models □ Model …**
Navigate to the working directory and select *s27_scan.v* from the list of files.
Click ***Run***



Build Model

Select Files/Directories

Directory : /afs/ece.cmu.edu/usr/cnigh/project5/cet_workdirs/s27_scan

Filter : *

- tbdata
- testresults
- s27_scan.v

Design Source

/afs/ece.cmu.edu/usr/cnigh/project5/cet_workdirs/s27_scan/s27_scan.v

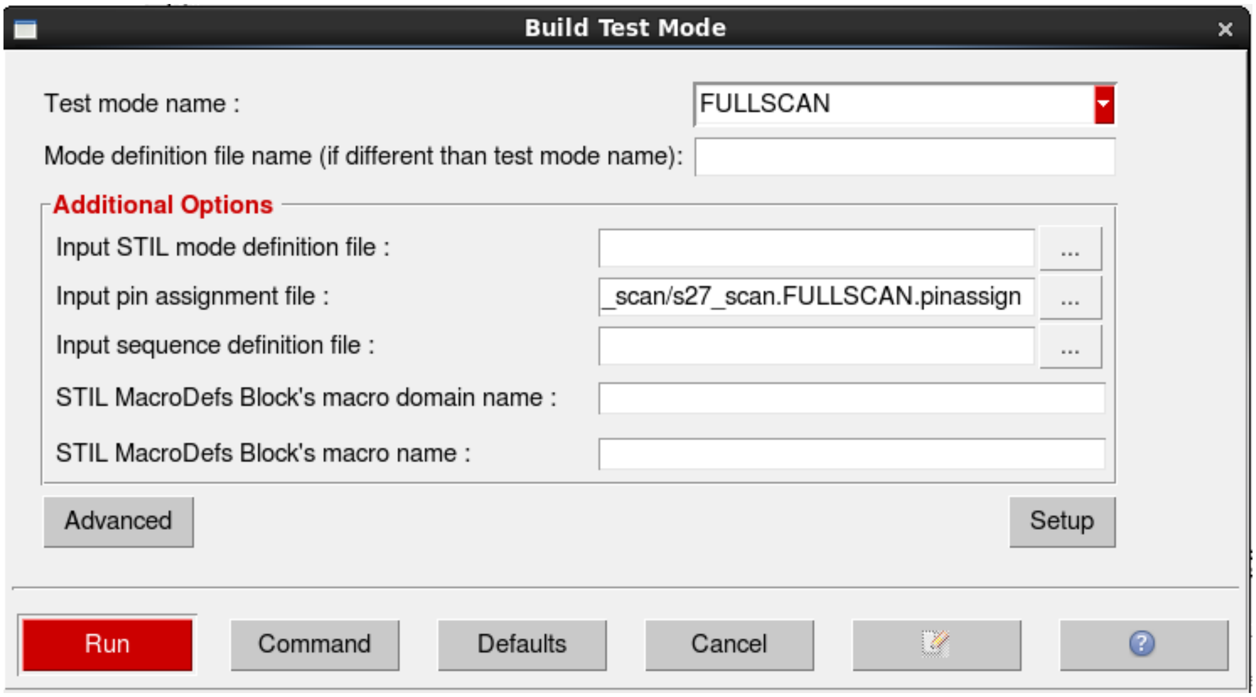
Cell Name

□ Perform additional checks for True-Time delay test

□ Allow Missing Cell Definitions

In a terminal, move (or copy) the file *s27_scan.FULLSCAN.pinassign* from `project5/p5example_s27/` to the project's working directory `project5/cet_workdirs/s27_scan/`. This simple file defines the clock signals used for any sequential circuits. You may wish to view this file, as you will need to create a pin assignment file for your own Project 5 DFT circuit.

Select **Verification** □ **Build Models** □ **Test Mode …**
Type “FULLSCAN” in for *Test mode name*
For *Input pin assignment file* select “s27_scan.FULLSCAN.pinassign” from your working directory `project5/cet_workdirs/s27_scan`.
Click ***Run***


Build Test Mode

Test mode name : FULLSCAN

Mode definition file name (if different than test mode name):

Additional Options

Input STIL mode definition file :

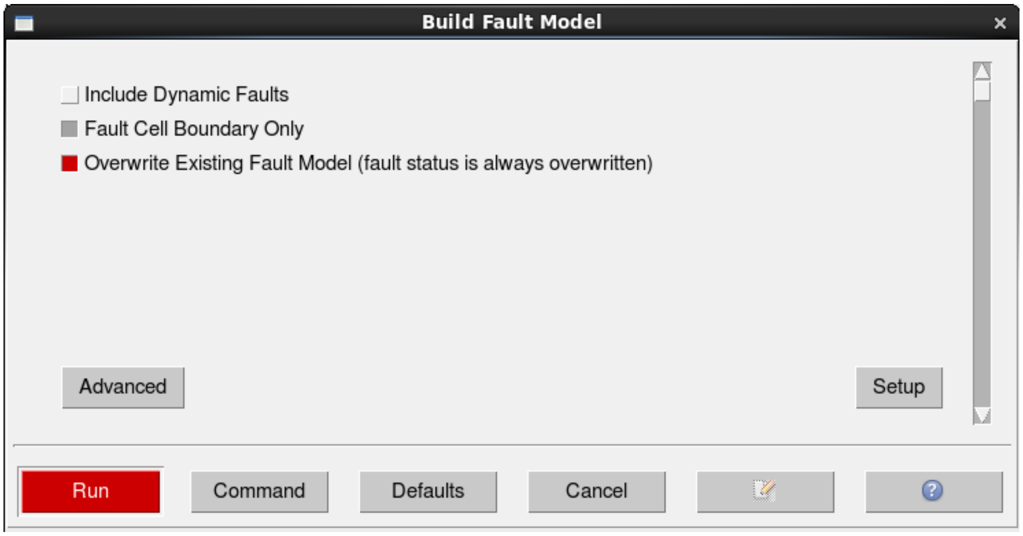
Input pin assignment file : _scan/s27_scan.FULLSCAN.pinassign

Input sequence definition file :

STIL MacroDefs Block's macro domain name :

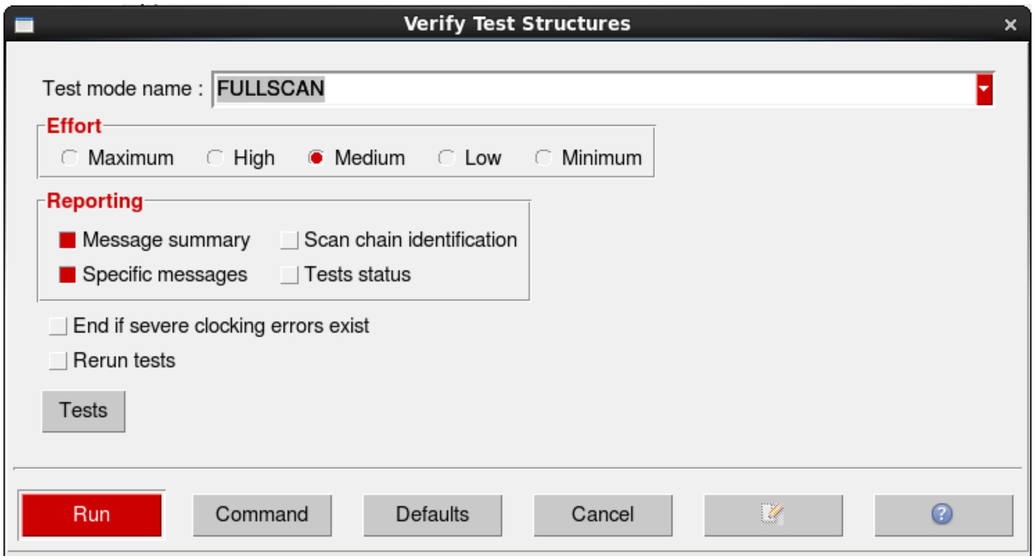
STIL MacroDefs Block's macro name :

Select **Verification** □ **Build Models** □ **Build Fault Model …**
Deselect *Include Dynamic Faults*
Click ***Run***


Build Fault Model

□ Include Dynamic Faults
▨ Fault Cell Boundary Only
■ Overwrite Existing Fault Model (fault status is always overwritten)

Select **Verification □ Verify □ Test Structures …**
Select "FULLSCAN" for the *Test mode name*
Click ***Run***



Test mode name : FULLSCAN

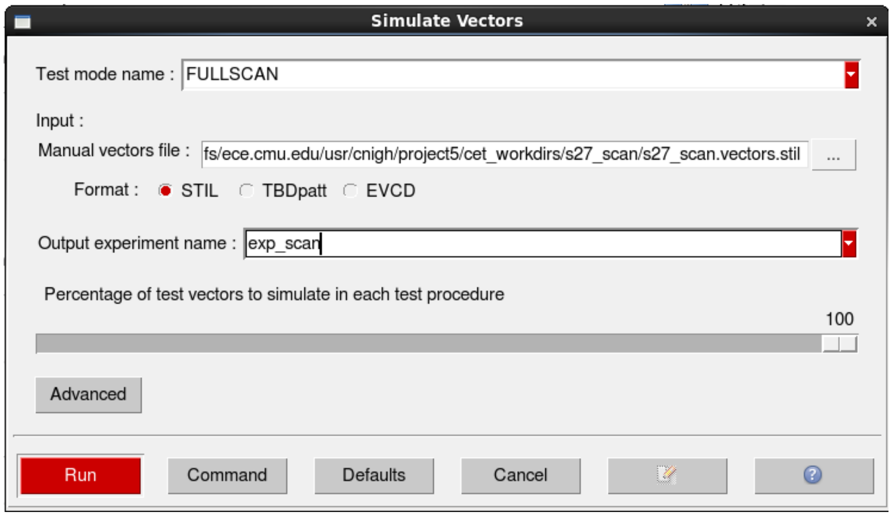
Effort
○ Maximum ○ High ● Medium ○ Low ○ Minimum

Reporting
■ Message summary □ Scan chain identification
■ Specific messages □ Tests status

□ End if severe clocking errors exist
□ Rerun tests

In a terminal, move (or copy) the *p5convert.py* created STIL files *s27_scan.vectors.stil* and *s27_scan.signals.stil* from `project5/p5example_s27/s27vec_temp/` to the project's working directory `project5/cet_workdirs/s27_scan/`.

Select **Tools □ Vectors □ Simulate …**
Select "FULLSCAN" for the *Test mode name*
For the *Manual vectors file*, select *s27_scan.vectors.stil* from the project's working directory
`project5/cet_workdirs/s27_scan/`
Type "exp_scan" for the *Output experiment name*
Select "STIL" for the *Format*



Test mode name : FULLSCAN

Input :

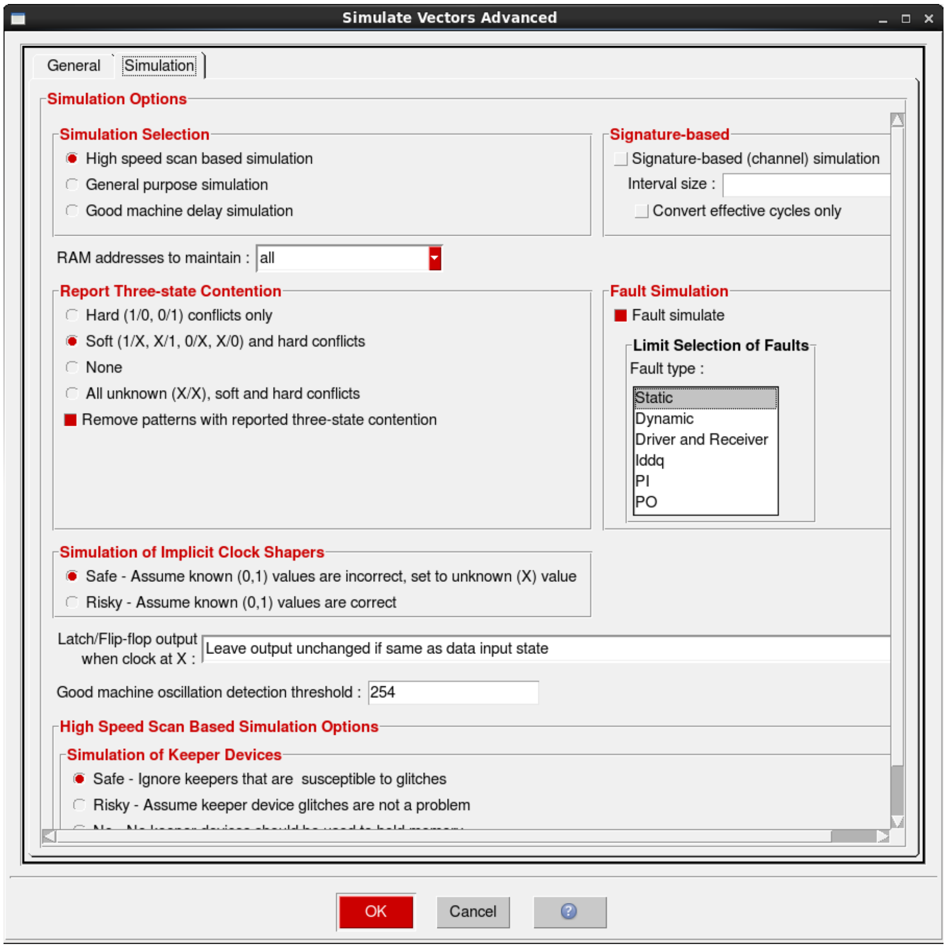
Manual vectors file : fs/ece.cmu.edu/usr/cnigh/project5/cet_workdirs/s27_scan/s27_scan.vectors.stil

Format : ● STIL ○ TBDpatt ○ EVCD

Output experiment name : exp_scan

Percentage of test vectors to simulate in each test procedure
100

Click ***Advanced***
Move to the *Simulation* tab
Under *Fault Simulation*, select “Static” as the *Fault type*



Simulate Vectors Advanced

General | Simulation

Simulation Options

Simulation Selection
- ● High speed scan based simulation
- ○ General purpose simulation
- ○ Good machine delay simulation

Signature-based
- ☐ Signature-based (channel) simulation
- Interval size :
- ☐ Convert effective cycles only

RAM addresses to maintain : all

Report Three-state Contention
- ○ Hard (1/0, 0/1) conflicts only
- ● Soft (1/X, X/1, 0/X, X/0) and hard conflicts
- ○ None
- ○ All unknown (X/X), soft and hard conflicts
- ☑ Remove patterns with reported three-state contention

Fault Simulation
- ☑ Fault simulate

Limit Selection of Faults
Fault type :
- Static
- Dynamic
- Driver and Receiver
- Iddq
- PI
- PO

Simulation of Implicit Clock Shapers
- ● Safe - Assume known (0,1) values are incorrect, set to unknown (X) value
- ○ Risky - Assume known (0,1) values are correct

Latch/Flip-flop output when clock at X : Leave output unchanged if same as data input state

Good machine oscillation detection threshold : 254

High Speed Scan Based Simulation Options

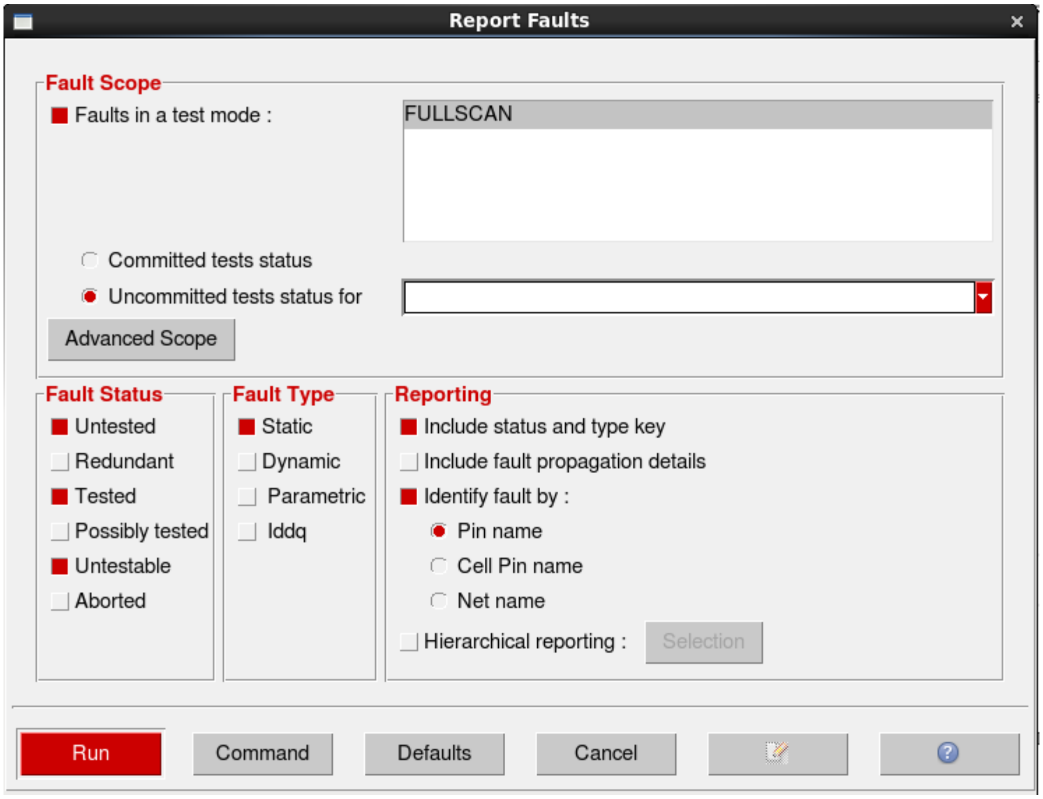
Simulation of Keeper Devices
- ● Safe - Ignore keepers that are susceptible to glitches
- ○ Risky - Assume keeper device glitches are not a problem
- ○ No - No keeper devices should be used to hold memory

Click ***OK*** to close this window

Click ***Run*** on the original window

The test vectors should obtain about 98% fault coverage. Only a certain set of untestable or newly-introduced faults will not be detected. All of the faults from the original combinational circuit will be covered. As a confirmation, you can report a list of every fault built by the fault model and its tested/untested status.
Looking through this report will also allow you to identify the undetected faults in your own Project 5 DFT circuit and test for them using additional test vectors or circuit modifications.

Select **Report** □ **Fault** □ **Logic Faults …**
Select *Faults in test mode*
Select “FULLSCAN”
Select *Uncommitted tests status* and “exp_scan”
Under *Fault Status*, select “Untested”, “Tested”, and “Untestable”.
Under *Reporting* select “Include status and type key” and “Identify fault by pin name”
Click ***Run***



Report Faults

Fault Scope

- ■ Faults in a test mode : FULLSCAN
- ○ Committed tests status
- ● Uncommitted tests status for

Advanced Scope

Fault Status

- ■ Untested
- □ Redundant
- ■ Tested
- □ Possibly tested
- ■ Untestable
- □ Aborted

Fault Type

- ■ Static
- □ Dynamic
- □ Parametric
- □ Iddq

Reporting

- ■ Include status and type key
- □ Include fault propagation details
- ■ Identify fault by :
  - ● Pin name
  - ○ Cell Pin name
  - ○ Net name
- □ Hierarchical reporting : Selection

The log file will also be created for this step, printing the same information that you see in the *Session Log*. To access this file, look for the file `testresults/logs/log_report_faults_FULLSCAN_exp_scan` in the working directory of your Encounter Test project.