University of Connecticut
Electrical and Computer Engineering

ECE 300: VLSI Design Verification and Testing
(Instructor: Mohammad Tehranipoor)

SYNOPSYS
A Quick Tool Setup for Synthesis & Test

DISCLAIMER: *The following steps aim at your setting SYNOPSYS toolset and running a sample VHDL program. SYNOPSYS is a large commercial CAD tool suite with many additional options that are not explained here. What we explain here is the minimum to run few tools. Nothing is guaranteed here. With this new version of SYNOPSYS that we installed, there might be problems in this document and some commands may not work exactly as explained. Proceed cautiously and use it at your own risk. I encourage you to assign enough time to familiarize yourself with this tool to be able to use it efficiently. Please report problems, corrections and suggestions about this document to* tehrani@engr.uconn.edu.

I. **Setup**

Most of the CAD tools, including SYNOPSYS, are accessible in various labs with UNIX/Linux workstations in ITE building. Many of these tools, including SYNOPSYS, provide shell commands to allow user run them without graphic user interface. This means that you can use SSH to one of these machines remotely and run them without GUI.

- You need to be familiar with the basic UNIX commands and one UNIX text editor (VI, EMACS, GVIM, EDIT, etc.)
- In order to run CAD tools in your UConn Engr UNIX account, make sure that your environment variables are set correctly:
  - For csh or tcsh users add the following line at the end of your **.login** file. Remember that in UNIX the files whose names start with a "." (e.g. **.login**) are hidden, to view them type "ls -a".
    source /apps/ecs-apps/software/synopsys/etc/cshrc.synopsys
  - For bash users add the following line at the end of your **.bash_profile** file.
    source /apps/ecs-apps/software/synopsys/etc/bashrc.synopsys

  You need to logout and login again, or do one of the following:
  source .login OR **source .bash_profile**

- The Synopsys On-line Documentation can be accessed with the command $sold from the command prompt.
- Make a directory in your home directory. For example, SYNOPSYS. Note that "˜ " refers to your root directory.
  mkdir ˜ /SYNOPSYS
- Within this directory you should have a directory called WORK. This directory is sometimes used by Synopsys tools to hold temporary values.
  mkdir ˜ /SYNOPSYS/WORK
- The Synopsys tool requires a shell setup file ˜ **/SYNOPSYS/.synopsys_dc.setup** to run **dc_shell** properly.

```
/* ------------------------------------------ */
/* Setup file to point to appropriate symbols */
/* and synthesis libraries.                   */
/* ------------------------------------------ */

company = "University of Connecticut" ;
designer = "First_name Last_name";
define_design_lib WORK -path ./WORK

search_path = {. /apps/ecs-apps/software/synopsys/etc} + search_path
link_library = {GSCLib_3.0.db};
target_library = {GSCLib_3.0.db};
symbol_library={class.sdb generic.sdb};
vhdlout_use_packages={IEEE.std_logic_1164};
vhdlout_write_components="false";
plot_command = "lpr -Pljsol"
```

- The Synopsys tool also requires a shell setup file ˜ **/SYNOPSYS/.synopsys_vss.setup** to run simulator properly.

```
/* ------------------------------------------ */
/* Setup file to define default data.         */
/* ------------------------------------------ */

WORK            > DEFAULT
DEFAULT         : ./WORK
TIMEBASE            = ns
```

II. **Preparation for Testing**

A. HDL Netlist Requirements

Designs provided in VHDL or Verilog format.

Example

```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.all;
ENTITY example IS
PORT(A, B, C, D, E: IN std_logic;
     F: OUT std_logic);
END;

ARCHITECTURE rtl OF example IS
SIGNAL w, x, y, z: std_logic;
BEGIN
w <= a AND b;
x <= c OR d;
y <= NOT e;
z <= w NOR x;
f <= y NAND z;
END;
```

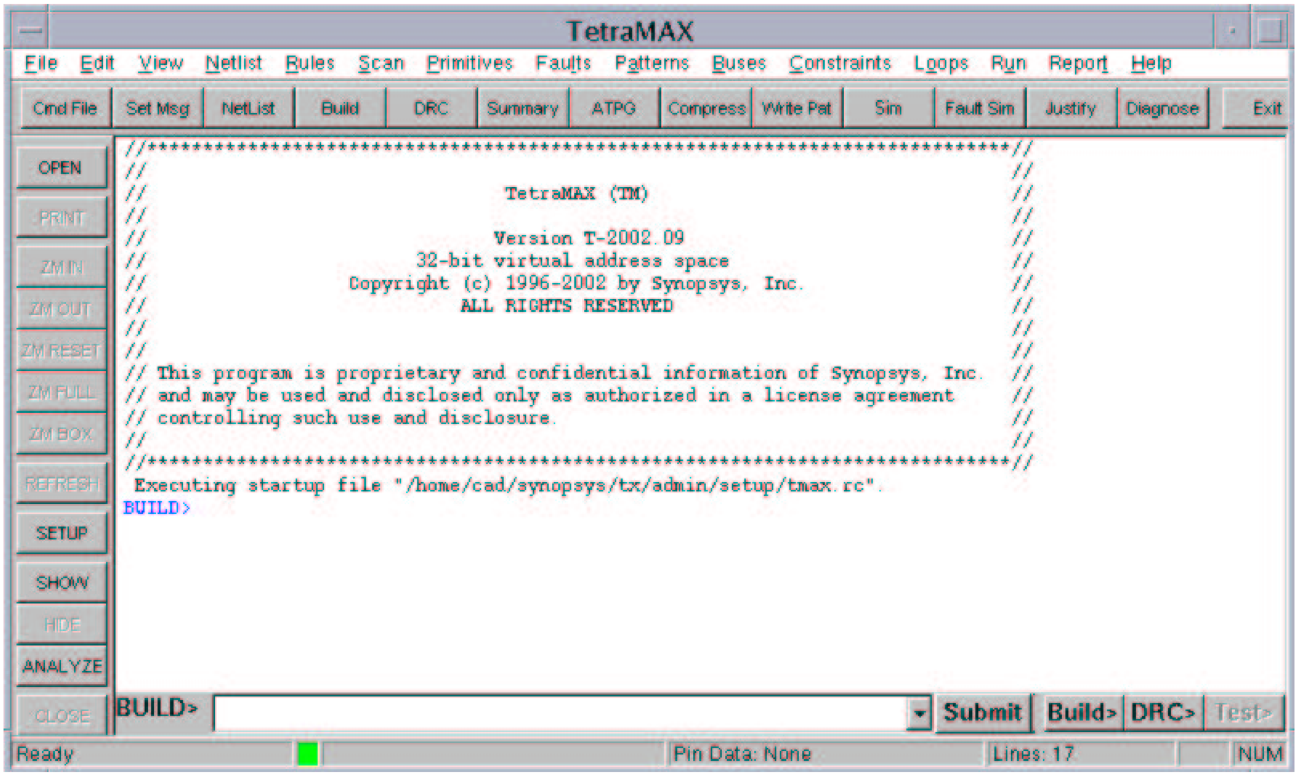
–OR–

Use the c17 Verilog benchmark from http://www.fm.vslib.cz/ kes/asic/iscas/

1 **<shell prompt> dc_shell-t**
2 **Read the HDL file in DFT Compiler.**
3 **Map and optimize to target library specified in .synopsys_dc.setup (GSCLib_3.0.db)**
3 **After synthesis save the design in Verilog format.**
4 **Save it as example_st.v.**
This will save your design in complete optimized structural model mapped to the target library.

B. TetraMAX Invocation

<shell prompt> tmax
This invokes the Graphical User Interface (GUI) for TetraMAX shown in Figure 1.


TetraMAX
File Edit View Netlist Rules Scan Primitives Faults Patterns Buses Constraints Loops Run Report Help
Cmd File Set Msg NetList Build DRC Summary ATPG Compress Write Pat Sim Fault Sim Justify Diagnose Exit
OPEN PRINT ZM IN ZM OUT ZM RESET ZM FULL ZM BOX REFRESH SETUP SHOW HIDE ANALYZE CLOSE
BUILD>
Submit Build> DRC> Test>
Ready
Pin Data: None
Lines: 17
NUM

```
//****************************************************************************//
//                                                                            //
//                             TetraMAX (TM)                                  //
//                                                                            //
//                            Version T-2002.09                               //
//                       32-bit virtual address space                         //
//                 Copyright (c) 1996-2002 by Synopsys, Inc.                  //
//                           ALL RIGHTS RESERVED                              //
//                                                                            //
//                                                                            //
// This program is proprietary and confidential information of Synopsys, Inc. //
// and may be used and disclosed only as authorized in a license agreement    //
// controlling such use and disclosure.                                       //
//                                                                            //
//****************************************************************************//
 Executing startup file "/home/cad/synopsys/tx/admin/setup/tmax.rc".
BUILD>
```

Figure 1. TetraMAX startup window.

C. Read Library Models

You can read the library models using the Read Netlist dialog box. Click the Netlist button in the command toolbar at the top of the TetraMAX main window.

1 **Copy the file containing the verilog models of the components “GSCLib_3.0.v’ into your working directory.** It can be found in the path ’/apps/ecs-apps/software/synopsys/etc/’.
2 **Click Netlist button in the command toolbar.**
3 **Select the GSCLib_3.0.v file.**
4 **Check Libary modules.**
5 **Click RUN.**

D. Read HDL File

You can read the netlist using the Read Netlist dialog box, or you can enter the *read netlist* command from the command line.

1 **Select the file, example_st.v**
2 **Click RUN.**

The following shows the *read netlist* command result.

```
----------------------------------------------------------------------
BUILD> read netlist ~/SYNOPSYS/tmax/example_st.v
 Begin reading netlist ( ~/SYNOPSYS/tmax/example_st.v )...
 External packages: ieee.std_logic_1164. Bit types: std_logic.
 End parsing VHDL file ./example.vhd with 0 errors;
 End reading netlist: #modules=1, top=example, #lines=20, CPU_time=0.02 sec,
 Memory=0MB
----------------------------------------------------------------------
```

E. Building the ATPG Model

Building the ATPG design model takes those parts of the design that are to be part of the ATPG process and removes the hierarchy. You can build the ATPG model for your design using the Run Build Model dialog box, or you can use the *run build_model* command from the command line.

1 **Click Build button in command toolbar.**
2 **Click RUN.**

The following shows the result of a build run.

```
--------------------------------------------------------------
 BUILD> run build_model example
 -------------------------------------------------------------
 Begin build model for topcut = example ...
 -------------------------------------------------------------
 There were 0 primitives and 0 faultable pins removed during model
 optimizations
 End build model: #primitives=11, CPU_time=0.02 sec, Memory=0MB
------------------------------------------------------------
 Begin learning analyses...
 End learning analyses, total learning CPU time=0.02 sec.
 -------------------------------------------------------------
```

The Graphical Schematic Viewer (GSV) toolbar is used to display the schematic view of the circuit. Click **SHOW** → **ALL**.It shows the schematic view of the circuit.

F. Perform Design Rule Check

You can perform DRC using the Run DRC dialog box, or you can execute the *run drc* command from the command line.

1 **Click the DRC button.**
2 **Click RUN.**

The result of a typical DRC run is shown below,

```
 DRC> run drc
 -------------------------------------------------------------
 Begin scan design rules checking...
 -------------------------------------------------------------
 Begin simulating test protocol procedures...
 Test protocol simulation completed, CPU time=0.00 sec.
 ------------------------------------------------------------
 Begin scan chain operation checking...
 Scan chain operation checking completed, CPU time=0.00 sec.
 ------------------------------------------------------------
 Begin nonscan rules checking...
 Nonscan cell summary: #DFF=0  #DLAT=0  tla_usage_type=none
 Nonscan rules checking completed, CPU time=0.00 sec.
 ------------------------------------------------------------
 Begin DRC dependent learning...
 Fast-sequential depth results: control=0(0), observe=0(0),
 detect=0(0), CPU time=0.00 sec
 DRC dependent learning completed, CPU time=0.00 sec.
 ------------------------------------------------------------
 DRC Summary Report
 ------------------------------------------------------------
 No violations occurred during DRC process.
 Design rules checking was successful, total CPU time=0.01 sec.
 ------------------------------------------------------------
```

III. **Perform ATPG**

1 **Click the ATPG button.**
Make sure, the pattern source is *internal* and *Add all faults* is selected for fault source.

2 **Click RUN.**

The result of a ATPG run is shown below,

```
-------------------------------------------------------------------
TEST> remove faults -all
 0 faults were removed from the fault list.
TEST> add faults -all
 40 faults were added to fault list.
TEST> run atpg
 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=40, abort_limit=10...
 7              40      0         0/0/0   100.00\%      0.02

     Uncollapsed Stuck Fault Summary Report
 -----------------------------------------------
 fault class                     code   #faults
 ------------------------------  ----  ---------
 Detected                         DT         40
 Possibly detected                PT          0
 Undetectable                     UD          0
 ATPG untestable                  AU          0
 Not detected                     ND          0
 -----------------------------------------------
 total faults                                40
 test coverage                           100.00\%
 -----------------------------------------------
            Pattern Summary Report
 -----------------------------------------------
 #internal patterns                           7
     #basic_scan patterns                     7
 -----------------------------------------------
```

A. Report Faults

1 **Faults → Report Faults...**
2 **Select Report Type as All.**
3 **Click OK.**

To report collaped faults, check the option **Report Collapsed**.

B. Report Patterns

1 **Patterns → Report Patterns...**
2 **Select Report Type as All.**
3 **Click OK.**

C. Pattern Compression

1 **Click the Compress button.**
2 **Click OK.**

The following shows the result of the test pattern compression.

```
-----------------------------------------------------------------
TEST> run pattern_compression 1 -min_eliminated_pats 0 -max_useless_passes
off -verbose

 Pass 1: Reverse order pattern compression performed on 7 patterns.
 -------------------------------------------
 #patterns     #faults     test      process
 stored     detect/active  coverage  CPU time
 ---------  -------------  --------  --------
 6              40      0   100.00\%      0.00
 Compression pass 1 completed: #patterns_deleted=1, CPU time=0.00
-----------------------------------------------------------------
```

D. Format Vectors for ATPG

We must convert TetraMAX representation of the vectors to a readable format.

1 **Click the Write Pat button.**
2 **Fill the output file name.**
3 **Select the file format as WGL.**
4 **Click OK.**

A WGL file, patterns.wgl is created in the directory of invocation.

E. Fault Simulation

1 **Click Fault Sim.**
2 **Select External in Pattern Source box and load WGL file.**
3 **Click Run.**

Fault simulation will report the fault coverage.

IV. **Procedure 2**

This procedure does not require the component models in a verilog file. All the primitives used in the top level module can be described at the begining of the vhdl file as shown below. Another method is to describe all the primitives in a separate file and read it before the top level module. The same procedure described in the previous sections can be followed after reading the netlist.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_components.all;
--------------------------------------
entity and is
  port ( A, B : in  std_logic;
    Z    : out std_logic);
```

```
end and;
architecture beh_and of and is
begin  -- beh_and
  Z <= A and B;
end beh_and;
--------------------------------------
entity or is
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end or;
architecture beh_or of or is
begin  -- beh_or
  Z <= A or B;
end beh_or;
--------------------------------------
entity not is
  port ( A : in  std_logic;
    Z    : out std_logic);
end not;
architecture beh_not of not is
begin  -- beh_not
  Z <= not A;
end beh_not;
--------------------------------------
entity nand is
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end nand;
architecture beh_nand of nand is
begin  -- beh_nand
  Z <= A nand B;
end beh_nand;
--------------------------------------
entity nor is
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end nor;
architecture beh_nor of nor is
begin  -- beh_nor
  Z <= A nor B;
end beh_nor;
--------------------------------------

--TOP LEVEL MODULE

ENTITY example IS
PORT(A, B, C, D, E: IN std_logic;
     F: OUT std_logic);
END;

ARCHITECTURE rtl OF example IS

component and
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end component;

component or
```

```
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end component;

component not
  port ( A : in  std_logic;
    Z    : out std_logic);
end component;

component nand
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end component;

component nor
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end component;

  SIGNAL w, x, y, z: std_logic;

BEGIN

   S_2 : not port map( A => E, Z => y);
   S_3 : nor port map( A => w, B => x, Z => z);
   S_4 : nand port map( A => y, B => z, Z => F);
   S_0 : and port map( A => A, B => B, Z => w);
   S_1 : or port map( A => C, B => D, Z => x);

END;
```

V. ATPG for Sequential Designs

A. HDL Netlist Requirements

Example (Figure 8.5, Pg. 217)

Note: The D flip-flop entity does not have a reset pin.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
entity and2 is
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end and2;
architecture beh_and2 of and2 is
begin  -- beh_and2
  Z <= A and B;
end beh_and2;
--------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
entity or2 is
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end or2;
architecture beh_or2 of or2 is
begin  -- beh_or2
  Z <= A or B;
end beh_or2;
--------------------------------------
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
entity dff is
  port ( D, clk : in  std_logic;
    Q  : out std_logic);
end dff;
architecture beh_dff of dff is
begin  -- beh_dff
  process(clk,D)
    begin
      if (clk’event and clk = ’1’) then Q <= D;
      end if;
    end process;
end beh_dff;
--------------------------------------

--TOP LEVEL MODULE
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY example IS
PORT(A, clk, rst : IN std_logic;
     B: OUT std_logic);
END;

ARCHITECTURE rtl OF example IS

component and2
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end component;

component or2
  port ( A, B : in  std_logic;
    Z    : out std_logic);
end component;

component dff
  port ( D, clk : in  std_logic;
    Q  : out std_logic);
end component;

SIGNAL  x, y, z: std_logic;

BEGIN

   S_0 : and2 port map( A => A, B => y, Z => x);
   S_1 : dff port map( D => x, clk => clk, Q => y);
   S_2 : dff port map( D => A, clk => clk, Q => z);
   S_3 : or2 port map( A => y, B => z, Z => B);


END;
```

1. **<shell prompt> dc_shell-t**
2. **Read the file in DFT Compiler.**
3. **Select the dff entity. (***current_design* **dff)**
4. **Run** *compile* **at the command line.**
5. **Save the top-level design in Verilog format.**

6. **Save it as example_st.v.**

This will save your design in complete structural model.

B. *ATPG process*

1. **Read the verilog models of the components “GSCLib_3.0.v”**
2. **Read the file, example_st.v**
3. **click ′Enhanced Seq Modeling′.**
4. **Build the ATPG Model.**
5. **Perform DRC.**
6. **Run ATPG.**
7. **Click ′Enable Full-Seq ATPG′ in General ATPG settings frame.**
8. **Uncheck ′Random Fill′ in Full Sequential Settings frame.**

Follow the same procedure described in the ATPG prcess of combinational circuits earlier in the tutorial.

C. *ATPG Results*

```
TEST> run atpg
 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=30, abort_limit=10...
 0               0      0         0/2/0     0.00\%      0.00

 ------------------------------------------------------------
 Begin Full-Sequential ATPG for 30 uncollapsed faults ...
  --- abort limit : 10 seconds, NO BACKTRACK LIMIT
 ------------------------------------------------------------
 #patterns  #faults        #ATPG faults  test       process
 stored     detect/active  red/au/abort  coverage   CPU time
 ---------  -------------  ------------  --------  ----------
 1              10     20         0/0/0   45.00\%         0.01
 2               6     14         0/0/0   65.00\%         0.02
 2               0     14         0/0/1   65.00\%        10.07
 2               0      2         0/9/1   65.00\%        10.08
 2 faults were identified as detected by implication,
TEST  COVERAGE is now 68.33\%.

     Uncollapsed Stuck Fault Summary Report
 -----------------------------------------------
 fault class                     code   #faults
 ------------------------------  ----  ---------
 Detected                         DT         18
 Possibly detected                PT          5
 Undetectable                     UD          0
 ATPG untestable                  AU          6
 Not detected                     ND          1
 -----------------------------------------------
 total faults                                30
 test coverage                            68.33\%
 -----------------------------------------------
            Pattern Summary Report
 -----------------------------------------------
 #internal patterns                           2
     #full_sequential patterns                2
 -----------------------------------------------
```

D. Pattern for fault A s-a-1

```
********************************
FAULT SITE = A
FAULT TYPE = s-a-1
********************************

PATTERN GENERATED BY ATPG

Pattern 0 (full_sequential)
 Time 0: period = 100
 Time 0: force_all_pis =   00
 Time 140: measure_all_pos =X
 Time 200: force_all_pis = X1
 Time 340: measure_all_pos =0

********************************
```