CoE 113 Machine Exercise 1

Tool Familiarization: Synopsys VCS & Design Compiler

1 Introduction

In this exercise, you will be taken through the process of using the Synopsys Verilog Compiler and Simulator (VCS) for compiling and simulating Verilog HDL code, as well as viewing the timing diagram of the simulation through DVE. You will also use Synopsys Design Compiler to synthesize your designs and map it to a generic 90nm library. Synthesis involves translation of high-level HDL constructs into a gate-level netlist. You will simulate your synthesized design to verify that it is still functionally equivalent to your RTL (unsynthesized) design.

2 Tool Familiarization

2.1 Setting up the Environment

1. In order to run VCS, we must first set up the necessary UNIX environment variables. Open a text editor and create a file named *set_synopsys.sh*. This will be your set-up script and should be executed every time you open a new terminal for use with VCS and the synthesis tool. The contents of the file should be the following:

```
#!/bin/bash
## License
export LM_LICENSE_FILE=27000@10.158.16.12
## VCS
export VCS_HOME=/eeei/tools/synopsys/I-2014.03
export VCS_BIN=/eeei/tools/synopsys/I-2014.03/bin
## SYN
export SYN_BIN=/eeei/tools/synopsys/M-2016.12-SP5-2/bin
## PATH
export PATH=$VCS_BIN:$SYN_BIN:$PATH
```

2. Run the script you created in the command line to set up the necessary environment variables. Make sure that you run the following commands in the same directory where you created the *set_synopsys.sh* script.

```
[uLab@artanis ~]$ source set_synopsys.sh
[uLab@artanis ~]$
```

3. To make sure that all necessary environment variables have been set properly, run the following commands. The terminal should display the values set in the script file.

```
[uLab@artanis ~]$ echo $LM_LICENSE_FILE
27000@10.158.16.12
[uLab@artanis ~]$ echo $VCS_HOME
/eeei/tools/synopsys/I-2014.03
[uLab@artanis ~]$ echo $VCS_BIN
/eeei/tools/synopsys/I-2014.03/bin
[uLab@artanis ~]$ echo $SYN_BIN
/eeei/tools/synopsys/M-2016.12-SP5-2/bin
[uLab@artanis ~]$
```

2.2 Using VCS for Compilation and Simulation

1. Now that we have already set up the necessary environment variables, we can now proceed to using the tools for compilation. Using the terminal where you ran the set-up script, unpack the *CoE113_ME1.tar* file you downloaded from the class webpage. Proceed to the *CoE113_ME1* directory. This directory should contain the following subdirectories:

   (a) *rtl* - This is where the RTL source Verilog files are placed. For this exercise, the source Verilog file has already been prepared.

   (b) *sim* - This is where the testbench source Verilog files are placed. This is also where simulations will be conducted. The testbench for the RTL source in the *rtl* directory has already been prepared for this exercise.

   (c) *cons* - This is where the constraints scripts are placed. The constraints file *timing.con* has already been prepared for this exercise.

   (d) *lib* - This is where the library-related files are placed. Four files from the Synopsys generic 90nm library have already been placed in this directory. (These files may have to be obtained through a symbolic link provided by your instructor.)

   (e) *mapped* - This is where the synthesized design will be placed.

   (f) *logs* - This is where the output log files from the synthesis process will be placed.

```
[uLab@artanis ~]$ tar -xvf CoE113_ME1.tar
[uLab@artanis ~]$ cd CoE113_ME1/
[uLab@artanis CoE113_ME1]$ ls
cons  lib  logs  mapped  rtl  sim
[uLab@artanis CoE113_ME1]$ ls cons
timing.con
[uLab@artanis CoE113_ME1]$ ls lib
saed90nm.sdb saed90nm.v saed90nm_typ.db saed90nm_typ.lib
[uLab@artanis CoE113_ME1]$ ls rtl
updown.v
[uLab@artanis CoE113_ME1]$ ls sim
tb_updown.v
[uLab@artanis CoE113_ME1]$
```

2. To use VCS for both compilation and simulation, your source Verilog files should contain at least two modules: the design to be tested and the testbench. Both modules can be placed in a single file or in separate files. In this exercise, the design to be tested is placed in the *CoE113_ME1/rtl/updown.v* file while the testbench is placed in the *CoE113_ME1/sim/tb_updown.v* file. From the *CoE113_ME1* directory, go to the *sim* directory. Your present working directory should now be *CoE113_ME1/sim.* Once inside the *sim* directory, compile the files mentioned earlier (the design to be tested and the testbench) by running the following command. Note that relative to your present working directory (*CoE113_ME1/sim*), the design file is located at *../rtl* whereas the testbench file is in the present working directory (*CoE113_ME1/sim*). If the command line flags an error that says the `vcs` command is not found, make sure that you have properly set up the environment variables before running the command again.

```
[uLab@artanis CoE113_ME1]$ cd sim
[uLab@artanis sim]$ vcs ../rtl/updown.v tb_updown.v -full64 -debug_pp
                         Chronologic VCS (TM)
         Version I-2014.03_Full64 -- Sat Jan 18 12:40:47 2020
               Copyright (c) 1991-2014 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.

Parsing design file '../rtl/updown.v'
Parsing design file 'tb_updown.v'
                                        .
```

```
                                   .
                                   .
../simv up to date
CPU time: .096 seconds to compile + .046 seconds to elab + .154 seconds to link
[uLab@artanis sim]$
```

3. If the compilation was successful, VCS should have created an executable simulation file named *simv* (by default) along with other files needed and generated by VCS. Run the *simv* file in the command line to execute the simulation.

```
[uLab@artanis sim]$ ls
csrc  simv  simv.daidir tb_updown.v
[uLab@artanis sim]$ ./simv
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version I-2014.03_Full64; Runtime version I-2014.03_Full64; Jan 18 12:42
     2020
VCD+ Writer I-2014.03_Full64 Copyright (c) 1991-2014 by Synopsys Inc.
$finish called from file "tb_updown.v", line 30.
$finish at simulation time           120000
          V C S   S i m u l a t i o n R e p o r t
Time: 120000 ps
CPU Time:     0.150 seconds;     Data structure size: 0.0Mb
Sat Jan 18 12:42:19 2020
[uLab@artanis sim]$
```

2.3 Using DVE for Viewing Simulation Results

1. If the simulation was successful, a waveform dumpfile named *tb_updown.vpd* should have been generated. To view this waveform, run DVE from the command line. This will open up a GUI window for the tool, as shown in Figure 1.

```
[uLab@artanis sim]$ ls
csrc  simv  simv.daidir tb_updown.v tb_updown.vpd ucli.key
[uLab@artanis sim]$ dve -full64 &
[1] 28109
[uLab@artanis sim]$
```

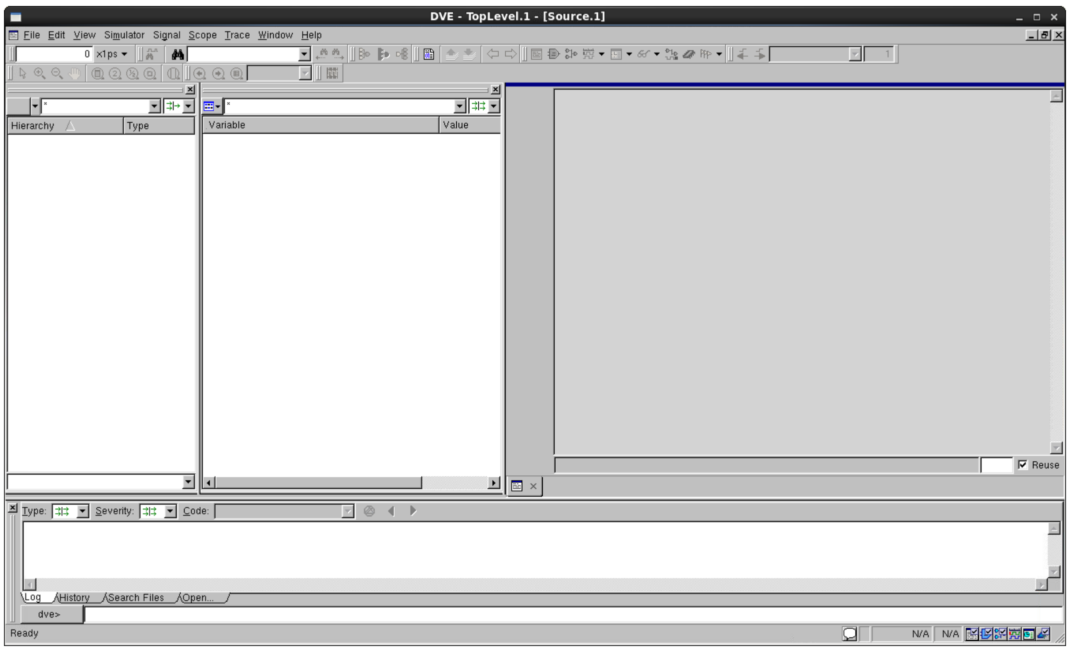
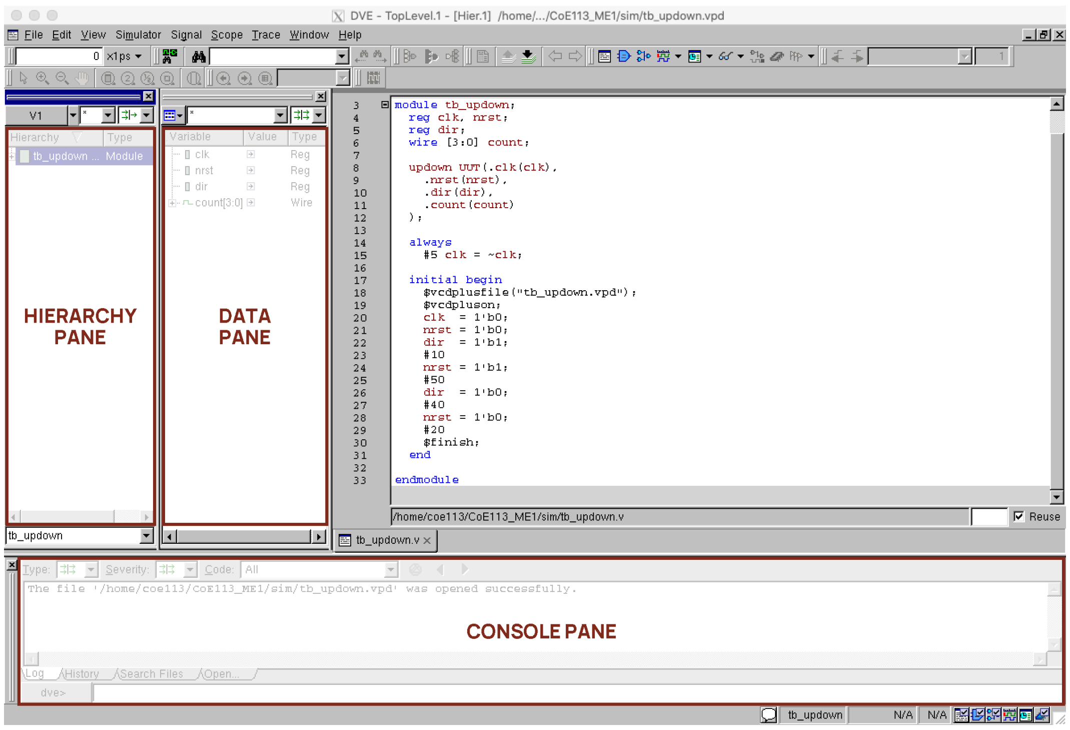


Figure 1: DVE graphical user interface.

2. On the main toolbar, click File then select “Open Database” (or press Ctrl+O). Select the dumpfile (*tb_updown.vpd*) and click “Open” to load the dumpfile into DVE.

3. Figure 2 shows the DVE GUI after loading the dumpfile. There are three panes of interest shown. The hierarchy pane shows the hierarchy of modules of the compiled Verilog files. The data pane shows the variables, signals, and registers of the currently selected module in the hierarchy pane. The console pane shows a console that can be used to execute DVE commands. Do not worry if you cannot find all three panes. You can toggle them into view by selecting Window → Panes in the main toolbar.



```
module tb_updown;
  reg clk, nrst;
  reg dir;
  wire [3:0] count;

  updown UUT(.clk(clk),
    .nrst(nrst),
    .dir(dir),
    .count(count)
  );

  always
    #5 clk = ~clk;

  initial begin
    $vcdplusfile("tb_updown.vpd");
    $vcdpluson;
    clk  = 1'b0;
    nrst = 1'b0;
    dir  = 1'b1;
    #10
    nrst = 1'b1;
    #50
    dir  = 1'b0;
    #40
    nrst = 1'b0;
    #20
    $finish;
  end

endmodule
```

```
/home/coe113/CoE113_ME1/sim/tb_updown.v
```

```
The file '/home/coe113/CoE113_ME1/sim/tb_updown.vpd' was opened successfully.
```

Figure 2: DVE GUI with loaded dumpfile.

4. To view the signals of the testbench, select *tb_updown* in the hierarchy pane, then select all four signals (`clk`, `nrst`, `dir`, and `count`) in the data pane by holding the control key and clicking on each signal. While selected, right-click anywhere on the GUI to pop up a menu and select “Add to waves” → “New wave view”. A new window will open similar to the one shown in Figure 3. This waveform viewer window will show a timing diagram of the simulation dumped into the dumpfile. A signal pane will display all signals that were selected. You can add additional signals to the waveform viewer by going back to the main DVE GUI, selecting a signal, right-clicking to bring up the menu, and selecting “Add to waves” → “Add to Wave.X” where Wave.X is the name of the waveform viewer window (alternatively, you can just select the signals and press Ctrl+4).

5. Also present in the waveform viewer is the reference marker, which can be moved around the timing diagram by simply clicking on the timing diagram or waveform. The signal window shows the respective values of the signals in the timing diagram at the time marked by the reference marker. You can zoom in and out of the waveform by using the zoom tools (Zoom Full, Zoom In 2x, Zoom Out 2x). You can also navigate through the waveform using the arrow buttons and sliding the navigation bar located at the bottom of the timing diagram. The ruler on the top of the waveform shows the time axis of the waveform currently displayed, while the one at the bottom shows all time values for the entire simulation, not just the one currently displayed. You can zoom in to a specific part of the waveform displayed by clicking on the top ruler and dragging it to the left or right to select the region you wish to zoom in. You can also zoom to any part of the simulation (not just the one currently displayed by performing the same action on the bottom ruler. Thus, selecting the whole bottom ruler would just display the waveform of the whole simulation (similar to a Zoom Full). Show the simulation waveform to your instructor.

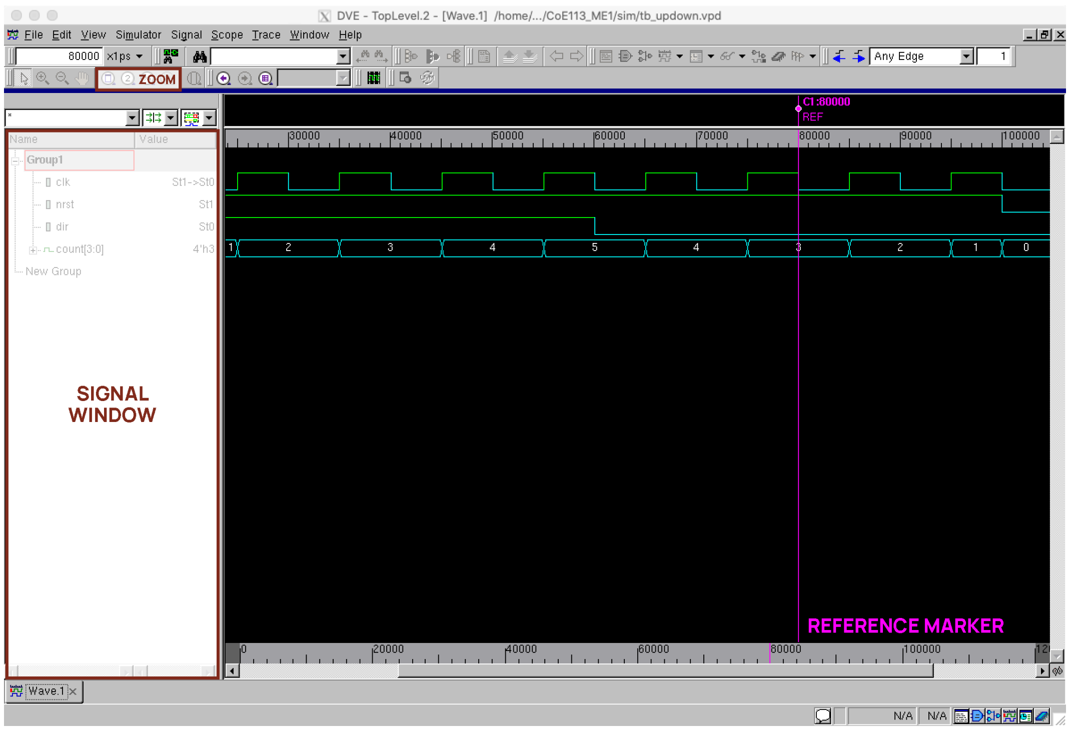

DVE - TopLevel.2 - [Wave.1] /home/.../CoE113_ME1/sim/tb_updown.vpd
File Edit View Simulator Signal Scope Trace Window Help
ZOOM
C1:80000
REF
Group1
clk
nrst
dir
count[3:0]
St1->St0
St1
St0
4'h3
SIGNAL WINDOW
REFERENCE MARKER
Wave.1

Figure 3: DVE waveform viewer.

2.4 Synthesizing the Design

1. Before running the synthesis tool, create a file named *.synopsys_dc.setup* using a text editor within the *CoE113_ME1* directory. This file is a startup script that will be executed by the Synopsys Design Compiler to set the search path within the program, as well as the target and link libraries. The contents of the file are shown below. These paths point to the necessary source Verilog files and libraries needed in synthesis.

```
set_app_var search_path "$search_path mapped lib cons rtl"
set_app_var target_library saed90nm_typ.db
set_app_var link_library "* $target_library"
```

2. Go to the *CoE113_ME1* directory and invoke DC. Make sure that you have all environment variables set properly before running DC.

```
[uLab@artanis CoE113_ME1]$ dc_shell -64bit

                        Design Compiler Graphical
                             DC Ultra (TM)
                              DFTMAX (TM)
                          Power Compiler (TM)
                             DesignWare (R)
                             DC Expert (TM)
                          Design Vision (TM)
                          HDL Compiler (TM)
                                  .
                                  .
                                  .
Initializing...
dc_shell>
```

3. After invoking DC, you should now be within the DC shell. The DC shell has its own commands but works similarly to BASH. Before proceeding any further, let us make sure that all the environment variables set by the *.synopsys_dc.setup* file have all been properly initialized. The *search_path* variable should include DC and synthesis-related installation directories, as well as the *mapped*, *lib*, *cons*, and *rtl*

directories in this exercise. Both the *target_library* and *link_library* variables point to the corresponding library files in the *lib* directory.

```
dc_shell> echo $search_path
. /eeei/tools/synopsys/M-2016.12-SP5-2/libraries/syn
    /eeei/tools/synopsys/M-2016.12-SP5-2/minpower/syn
    /eeei/tools/synopsys/M-2016.12-SP5-2/dw/syn_ver
    /eeei/tools/synopsys/M-2016.12-SP5-2/dw/sim_ver mapped lib cons rtl
dc_shell> echo $target_library
saed90nm_typ.db
dc_shell> echo $link_library
* saed90nm_typ.db
dc_shell>
```

4. Read the design files by running the `read_verilog` command. Although the design files are stored in the *rtl* directory, the *search_path* variable allows DC to search beyond the current directory. In this example, the whole design is contained in a single file. In cases where there are multiple source Verilog files in the design, you must perform `read_verilog` for each source file.

```
dc_shell> read_verilog updown.v
Loading db file '/home/coe113/CoE113_ME1/lib/saed90nm_typ.db'
Loading db file '/eeei/tools/synopsys/M-2016.12-SP5-2/libraries/syn/gtech.db'
Loading db file '/eeei/tools/synopsys/M-2016.12-SP5-2/libraries/syn/standard.sldb
    '
  Loading link library 'saed90nm_typ'
                                    .
                                    .
                                    .
Presto compilation completed successfully.
Current design is now '/home/coe113/CoE113_ME1/rtl/updown.db:updown'
Loaded 1 design.
Current design is 'updown'.
updown
dc_shell>
```

5. For this exercise, the design consists of only a single module, *updown.* In cases where the design is made up of more than a single module, you would need to use the `current_design` command to select the topmost module before synthesis.

```
dc_shell> current_design updown
Current design is 'updown'.
{updown}
dc_shell>
```

6. Link all modules and library components referenced in the current design using the `link` command. The purpose of this command is to locate all of the design and library components referenced in the current design and connect (or link) them to the current design. If linking is successful, the command will return a '1'.

```
dc_shell> link

  Linking design 'updown'
  Using the following designs and libraries:
  --------------------------------------------------------------------------
  updown                    /home/coe113/CoE113_ME1/rtl/updown.db
  saed90nm_typ (library)    /home/coe113/CoE113_ME1/lib/saed90nm_typ.db

1
dc_shell>
```

7. You may check the current design for consistency using the `check_design` command. This command will return a '1' when successful. This command also outputs a summary of errors or warnings if any arise.

```
dc_shell> check_design

****************************************
check_design summary:
Version:    M-2016.12-SP5-2
Date:       Sat Jan 18 12:58:51 2020
****************************************

                 Name                                           Total
--------------------------------------------------------------------------------
Cells                                                            2
   Cells do not drive (LINT-1)                                   2
--------------------------------------------------------------------------------

Warning: In design 'updown', cell 'C36' does not drive any nets. (LINT-1)
Warning: In design 'updown', cell 'C37' does not drive any nets. (LINT-1)
1
dc_shell>
```

8. Apply the timing constraints by running the *cons/timing.con* file. Timing constraints define the allowable clock frequency and tolerated delays for your design, and will be the basis for optimization. Similar to the previous commands, this step should return a '1' if successful.

```
dc_shell> source timing.con
1
dc_shell>
```

9. Check for possible timing problems in the current design using the `check_timing` command. This command outputs a summary of errors or warnings if any arise and will return a '1' if successful.

```
dc_shell> check_timing
Information: Changed wire load model for 'DW01_dec_width4' from '(none)' to '
    ForQA'. (OPT-170)
                                        .
                                        .
                                        .
Information: Checking partial_input_delay...
Warning: there are 3 input ports that only have partial input delay specified. (
    TIM-212)
--------------------
clk
nrst
dir
1
dc_shell>
```

10. Compile your design using the `compile` command.

```
dc_shell> compile
Information: Evaluating DesignWare library utilization. (UISN-27)

============================================================================
| DesignWare Building Block Library |      Version       | Available |
============================================================================
| Basic DW Building Blocks       | M-2016.12-DWBB_201612.5.1
                                                        |     *     |
| Licensed DW Building Blocks    |                      |           |
============================================================================
                                     .
                                     .
                                     .
```

```
                                       TOTAL
   ELAPSED          WORST NEG  SETUP    DESIGN
    TIME     AREA     SLACK    COST    RULE COST       ENDPOINT
  --------- --------- --------- --------- --------- -------------------------
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
    0:00:01    245.1     0.00      0.0       0.0
Loading db file '/home/coe113/CoE113_ME1/lib/saed90nm_typ.db'


Note: Symbol # after min delay cost means estimated hold TNS across all active
    scenarios


  Optimization Complete
  ---------------------
1
dc_shell>
```

11. To check if the compiled design violates any constraints, run the `report_constraint` command. This command will return a '1' if all constraints are met.

```
dc_shell> report_constraint -all_violators
Information: Updating design information... (UID-85)

****************************************
Report : constraint
        -all_violators
Design : updown
Version: M-2016.12-SP5-2
Date   : Sat Jan 18 13:01:12 2020
****************************************

This design has no violated constraints.

1
dc_shell>
```

12. To view an estimate of the area taken up by the design, run the `report_area` command. All numerical values shown in the report have units which are defined in the libraries.

```
dc_shell> report_area

****************************************
Report : area
Design : updown
Version: M-2016.12-SP5-2
Date   : Sat Jan 18 13:01:27 2020
****************************************

Library(s) Used:

    saed90nm_typ (File: /home/coe113/CoE113_ME1/lib/saed90nm_typ.db)
```

```
Number of ports:                    7
Number of nets:                    20
Number of cells:                   14
Number of combinational cells:     10
Number of sequential cells:         4
Number of macros/black boxes:       0
Number of buf/inv:                  1
Number of references:               9

Combinational area:        116.121600
Buf/Inv area:                5.529600
Noncombinational area:     129.024002
Macro/Black Box area:        0.000000
Net Interconnect area:       9.075486

Total cell area:           245.145602
Total area:                254.221088
1
dc_shell>
```

13. To view timing information of the synthesized design, run the `report_timing` command. All numerical values shown in the report have units which are defined in the libraries. In general, the report shows the amount of time allotted to the critical path (as specified in the timing constraints), the amount of delay in the critical path, and the "slack" or the timing margin you have for the data passing through the critical path. Successful synthesis requires that the slack is greater than zero, which means that the time it takes for the data to pass through the critical path is less than the allotted time.

```
dc_shell> report_timing

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : updown
Version: M-2016.12-SP5-2
Date   : Sat Jan 18 13:01:59 2020
****************************************

Operating Conditions: TYPICAL Library: saed90nm_typ
Wire Load Model Mode: enclosed

  Startpoint: dir (input port clocked by clk)
  Endpoint: count_reg[2]
            (rising edge-triggered flip-flop clocked by clk)
                                    .
                                    .
                                    .
  data required time                                8.41
  -----------------------------------------------------------
  data required time                                8.41
  data arrival time                                -3.28
  -----------------------------------------------------------
  slack (MET)                                       5.13


1
dc_shell>
```

14. Create a Standard Delay Format (SDF) back-annotation file of your design. This file will be used to provide delay information for simulation purposes. Delay information will be estimated from data specified in the libraries. In this example, the SDF file is written to the *mapped* directory.

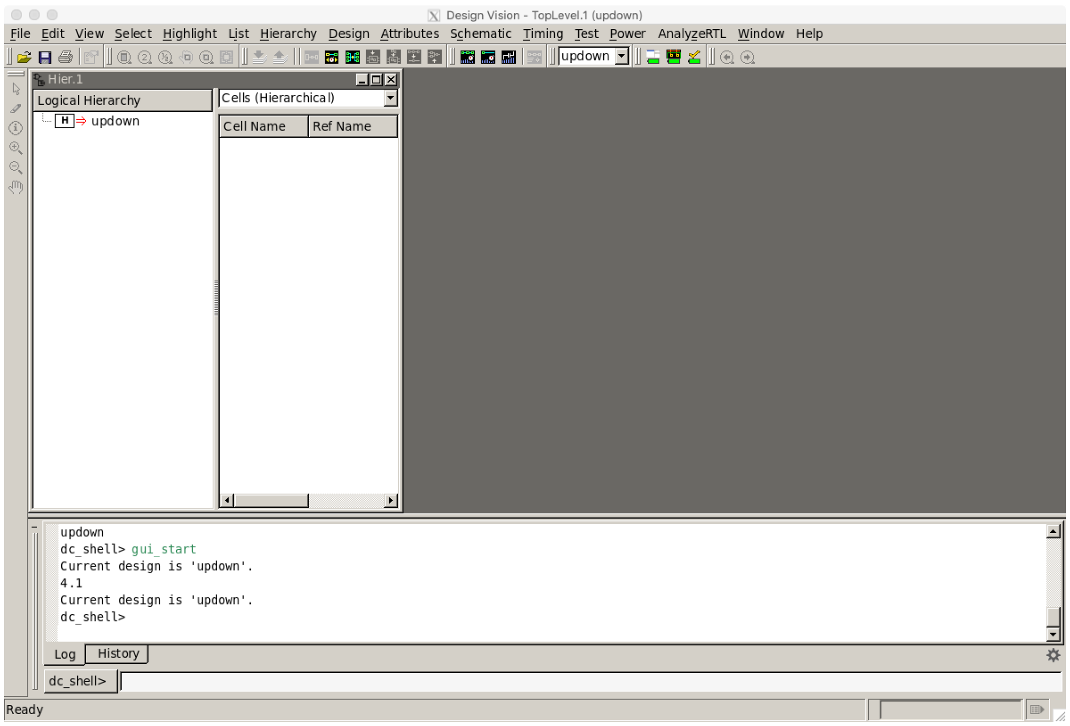
```
dc_shell> write_sdf -version 1.0 mapped/updown_mapped.sdf
Information: Annotated 'cell' delays are assumed to include load delay. (UID-282)
Information: Writing timing information to file '/home/coe113/CoE113_ME1/mapped/
    updown_mapped.sdf'. (WT-3)
1
dc_shell>
```

15. Write the synthesized design netlist from DC memory to a Verilog file using the `write` command. This file will be used later for simulation. In this example, the design is written to the *mapped* directory, signifying that this Verilog file is mapped to a specific technology library.

```
dc_shell> write -f verilog -hier -out mapped/updown_mapped.v
Writing verilog file '/home/coe113/CoE113_ME1/mapped/updown_mapped.v'.
1
dc_shell>
```

16. In order to view the schematic of the synthesized design, we must first invoke the GUI of DC. After invoking the GUI, a new window should appear similar to the one shown in Figure 4.

```
dc_shell> gui_start
dc_shell> Current design is 'updown'.
4.1
Current design is 'updown'.
dc_shell>
```

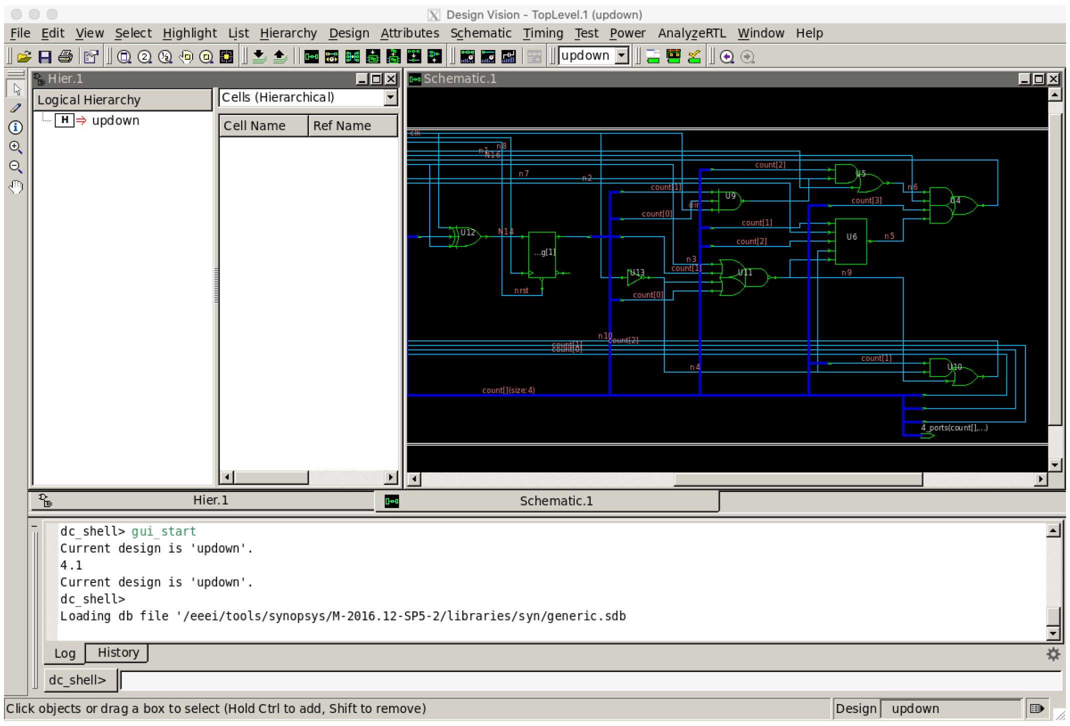

Design Vision - TopLevel.1 (updown)
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power AnalyzeRTL Window Help
updown
Hier.1
Logical Hierarchy
Cells (Hierarchical)
Cell Name
Ref Name
H
updown
Log
History
dc_shell>
Ready

```
updown
dc_shell> gui_start
Current design is 'updown'.
4.1
Current design is 'updown'.
dc_shell>
```

Figure 4: Design Compiler graphical user interface.

17. Select the *updown* module from the hierarchy pane and right-click. Select the "Schematic View" option by left-clicking on it (alternatively, you can also press Ctrl+G instead of bringing up this window).

18. The schematic of the synthesized design should now be shown in the GUI. Since we are still viewing the top-level module, descend into the schematic by double-clicking on the top-level module (similar to the controls of DVE). You should see a window similar to the one shown in Figure 5. Take note that you can also zoom in and out to get a better view of the schematic.


Design Vision - TopLevel.1 (updown)
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power AnalyzeRTL Window Help
Hier.1
Logical Hierarchy
updown
Cells (Hierarchical)
Cell Name
Ref Name
Schematic.1
Log
History
dc_shell>
Click objects or drag a box to select (Hold Ctrl to add, Shift to remove)
Design
updown

```
dc_shell> gui_start
Current design is 'updown'.
4.1
Current design is 'updown'.
dc_shell>
Loading db file '/eeei/tools/synopsys/M-2016.12-SP5-2/libraries/syn/generic.sdb
```

Figure 5: Schematic view of the synthesized design.

19. Exit DC by either typing `quit` on the command line or selecting exit on the GUI. Make sure that the synthesized design has been written out to a file by viewing the contents of *CoE113_ME1/mapped/updown_mapped.v* on a text editor. You should have a file similar to the one shown below. Notice that the mapped file is basically a netlist with instantiations of modules which are defined in the libraries (*saed90nm.v*).

```
module updown ( clk, nrst, dir, count );
  output [3:0] count;
  input clk, nrst, dir;
  wire  N14, N15, N16, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10;

  DFFARX1 \count_reg[0] ( .D(n3), .CLK(clk), .RSTB(nrst), .Q(count[0]), .QN(n3) );
  DFFARX1 \count_reg[1] ( .D(N14), .CLK(clk), .RSTB(nrst), .Q(count[1]) );
                                            .
                                            .
                                            .
  INVX0 U13 ( .INP(dir), .ZN(n4) );
endmodule
```

2.5 Functional Verification of Synthesized Design

1. To verify the functionality of the synthesized design, we simply need to compile the mapped file with the testbench instead of the original Verilog source file. Before compilation, annotate a timescale directive to the mapped Verilog file. Using a text editor, edit the *CoE113_ME1/mapped/updown_mapped.v* to append a timescale directive.

```
‘timescale 1ns/1ps

module updown ( clk, nrst, dir, count );
  output [3:0] count;
  input clk, nrst, dir;
  wire  N14, N15, N16, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10;

  DFFARX1 \count_reg[0] ( .D(n3), .CLK(clk), .RSTB(nrst), .Q(count[0]), .QN(n3) );
  DFFARX1 \count_reg[1] ( .D(N14), .CLK(clk), .RSTB(nrst), .Q(count[1]) );
```

```
                                    .
                                    .
                                    .
  INVX0 U13 ( .INP(dir), .ZN(n4) );
endmodule
```

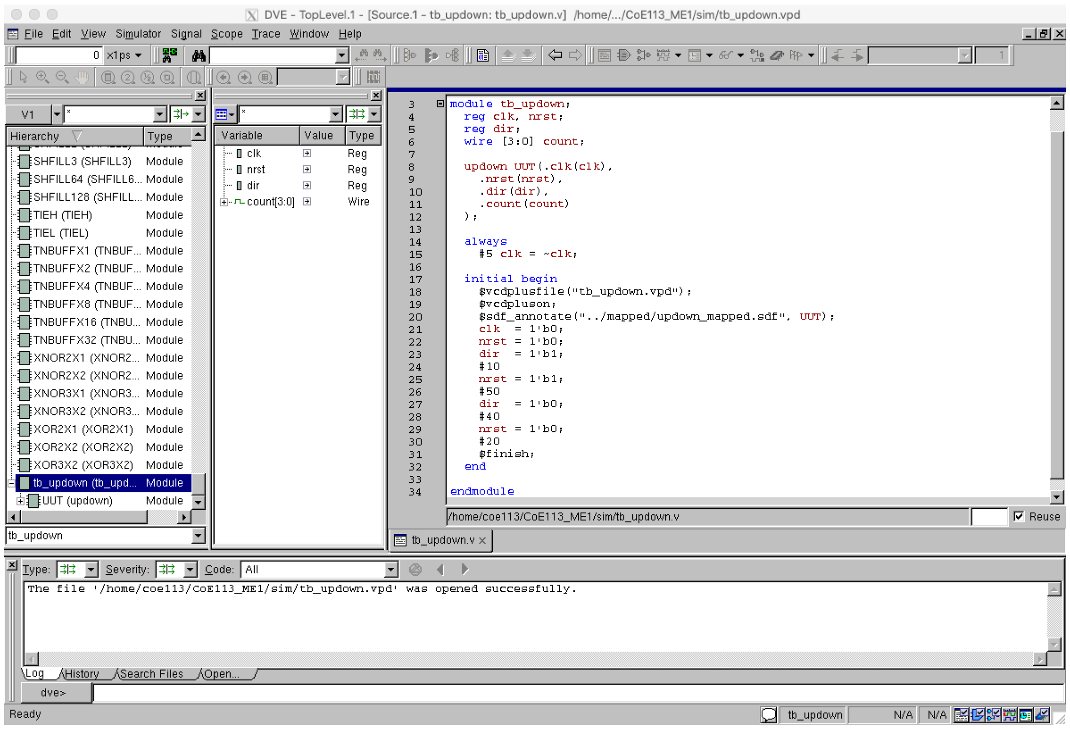
2. We also need to add additional timing information generated by the synthesis process to our simulation. In this particular exercise, that timing information is placed on the *mapped/updown_mapped.sdf* file. To add the timing information to our testbench, we will use the `$sdf_annotate` directive. Add the `$sdf_annotate` line to your testbench inside the stimulus generation part.

```
'timescale 1ns/1ps

module tb_updown;
                                    .
                                    .
                                    .
  initial begin
    $vcdplusfile("tb_updown.vpd");
    $vcdpluson;
    $sdf_annotate("../mapped/updown_mapped.sdf", UUT);
                                    .
                                    .
                                    .
    $finish;
  end

endmodule
```

3. Now that we have made the necessary modifications to our mapped Verilog design and our testbench, we can now proceed to compile and simulate our designs. Proceed to the *CoE113_ME1/sim* directory to compile and simulate the designs as shown. In compiling the Verilog files, make sure to include the testbench, mapped file, and standard cell library.

```
[uLab@artanis sim]$ vcs ../mapped/updown_mapped.v ../lib/saed90nm.v tb_updown.v
    -full64 -debug_pp +neg_tchk -R -l vcs.log
                       Chronologic VCS (TM)
        Version I-2014.03_Full64 -- Sat Jan 18 13:13:02 2020
              Copyright (c) 1991-2014 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.

Parsing design file '../mapped/updown_mapped.v'
Parsing design file '../lib/saed90nm.v'
Parsing design file 'tb_updown.v'
                                    .
                                    .
                                    .
VCD+ Writer I-2014.03_Full64 Copyright (c) 1991-2014 by Synopsys Inc.
$finish called from file "tb_updown.v", line 31.
$finish at simulation time           120000
         V C S   S i m u l a t i o n R e p o r t
Time: 120000 ps
CPU Time:     0.190 seconds;     Data structure size: 0.3Mb
Sat Jan 18 13:13:16 2020
CPU time: 2.685 seconds to compile + .100 seconds to elab + .349 seconds to link
    + .248 seconds in simulation
[uLab@artanis sim]$
```

4. Invoke DVE and load the *tb_updown.vpd* database. The hierarchy pane should show more than the usual amount of modules because even unused standard cells found in the library were also compiled. Navigate the hierarchy pane to show the *tb_updown* module and expand it, as shown in Figure 6.



DVE - TopLevel.1 - [Source.1 - tb_updown: tb_updown.v] /home/.../CoE113_ME1/sim/tb_updown.vpd

Hierarchy	Type
SHFILL3 (SHFILL3)	Module
SHFILL64 (SHFILL6...	Module
SHFILL128 (SHFILL...	Module
TIEH (TIEH)	Module
TIEL (TIEL)	Module
TNBUFFX1 (TNBUF...	Module
TNBUFFX2 (TNBUF...	Module
TNBUFFX4 (TNBUF...	Module
TNBUFFX8 (TNBUF...	Module
TNBUFFX16 (TNBU...	Module
TNBUFFX32 (TNBU...	Module
XNOR2X1 (XNOR2...	Module
XNOR2X2 (XNOR2...	Module
XNOR3X1 (XNOR3...	Module
XNOR3X2 (XNOR3...	Module
XOR2X1 (XOR2X1)	Module
XOR2X2 (XOR2X2)	Module
XOR3X2 (XOR3X2)	Module
tb_updown (tb_upd...	Module
UUT (updown)	Module

Variable	Value	Type
clk		Reg
nrst		Reg
dir		Reg
count[3:0]		Wire

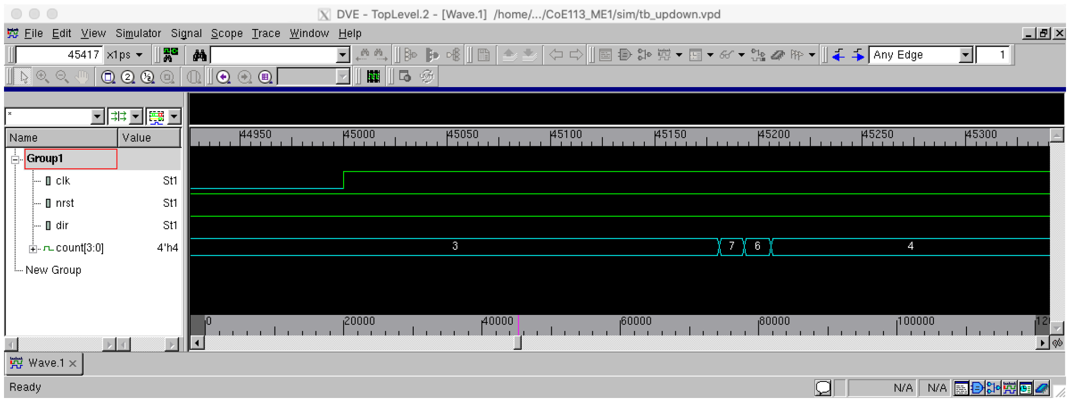
```
module tb_updown;
  reg clk, nrst;
  reg dir;
  wire [3:0] count;

  updown UUT(.clk(clk),
    .nrst(nrst),
    .dir(dir),
    .count(count)
  );

  always
    #5 clk = ~clk;

  initial begin
    $vcdplusfile("tb_updown.vpd");
    $vcdpluson;
    $sdf_annotate("../mapped/updown_mapped.sdf", UUT);
    clk  = 1'b0;
    nrst = 1'b0;
    dir  = 1'b1;
    #10
    nrst = 1'b1;
    #50
    dir  = 1'b0;
    #40
    nrst = 1'b0;
    #20
    $finish;
  end

endmodule
```

/home/coe113/CoE113_ME1/sim/tb_updown.v

```
The file '/home/coe113/CoE113_ME1/sim/tb_updown.vpd' was opened successfully.
```

Figure 6: Testbench database hierarchy pane.

5. Select all of the signals in the *tb_updown* module and display them in the waveform viewer. One important difference between the synthesized and unsynthesized designs is that glitches and logic delays are introduced in the synthesized design. Zooming in to a rising edge clock transition clearly shows this, as shown in Figure 7. Show this simulation waveform to your instructor.



DVE - TopLevel.2 - [Wave.1] /home/.../CoE113_ME1/sim/tb_updown.vpd

Name	Value
Group1	
clk	St1
nrst	St1
dir	St1
count[3:0]	4'h4
New Group	

Figure 7: Glitches and logic delay in synthesized design.

2.6 Automating the Synthesis Process Using Scripts

1. The synthesis process discussed previously can be automated by creating a script file and using it as an input argument to the Design Compiler. Delete the old *CoE113_ME1* directory and re-extract it so that you can repeat the synthesis procedure (this time using a script). Make sure that all necessary configuration files for Design Compiler are set properly.

2. Create a script file named *compile.tcl* using a text editor. Its contents should be the following:

```
read_verilog updown.v
current_design updown
link
check_design > logs/check_design.log
source timing.con
compile
report_constraint -all_violators > logs/constraint_report.log
report_area > logs/area_report.log
report_timing > logs/timing_report.log
write_sdf -version 1.0 mapped/updown_mapped.sdf
write -f verilog -hier -out mapped/updown_mapped.v
quit
```

3. Run `dc_shell` with the script file using the following command. Make sure that the paths required by your synthesis script are consistent with where you place and run the script.

```
[uLab@artanis CoE113_ME1]$ dc_shell -f compile.tcl -output_log_file
    logs/compile.log

                    Design Compiler Graphical
                          DC Ultra (TM)
                           DFTMAX (TM)
                                    .
                                    .
                                    .
Writing verilog file '/home/coe113/CoE113_ME1/mapped/updown_mapped.v'.
1
quit

Thank you...
[uLab@artanis CoE113_ME1]$
```

4. Design Compiler will automatically execute each command in the script file. For the `check_design` and `report` commands, all outputs were forwarded to text files (*check_design.log*, *constraint_report.log*, *area_report.log*, and *timing_report.log*). The rest of the outputs during execution will be placed in the text file named *compile.log*. All of these text files are placed in the *logs* subdirectory.