# Lab 2: Synthesis

August 24, 2015

## Introduction

For this lab exercise, you will 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 then simulate your synthesized design to verify that is still functionally equivalent to your RTL (unsynthesized) design.

## Work Directory Structure

1. Extract lab_02.tar and go to the lab_02 directory view the directory structure. On the main directory, you should be able to find the following subdirectories:

   - cons - This is where your constraints scripts are placed. The constraints file timing.con has already been prepared for this exercise.
   - lib - This is where 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)
   - rtl - This is where your RTL source Verilog files are placed. For this exercise, the source Verilog file has already been prepared.
   - sim - This is where 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.
   - mapped - This is where you will place the synthesized design.

   ```
   [snap@artanis ~]$ tar -xvf lab_02.tar
   [snap@artanis ~]$ cd lab_02
   [snap@artanis lab_02]$ ls
   cons  lib  mapped  rtl  sim
   [snap@artanis lab_02]$ ls cons
   timing.con
   [snap@artanis lab_02]$ ls lib
   saed90nm.sdb  saed90nm_typ.db  saed90nm_typ.lib  saed90nm.v
   [snap@artanis lab_02]$ ls rtl
   updown.v
   [snap@artanis lab_02]$ ls sim
   tb_updown.v
   [snap@artanis lab_02]$
   ```

2. Go to the lab_02/sim directory and verify the functionality of the counter described in the file rtl/updown.v. Remember, the design file is located at ../rtl relative to your present working directory (lab_02/sim directory) and the testbench is in the present working directory. Show your simulation waveform to your instructor.

3. Before running the synthesis tool, create a file named .synopsys_dc.setup using a text editor within the lab_02 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.

   ```
   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"
   ```

## Synthesizing the Design

1. Go to the lab_02 directory and invoke DC. Make sure that you have all environment variables set properly before running DC (refer to lab exercise 1).

```
[snap@artanis lab_02]$ source set_synopsys.sh
[snap@artanis lab_02]$ dc_shell -64bit
                      Design Compiler Graphical
                            DC Ultra (TM)
                             DFTMAX (TM)
                         Power Compiler (TM)
                           DesignWare (R)
                           DC Expert (TM)
                         Design Vision (TM)
                                 ...
Initializing...
dc_shell>
```

2. 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 target_library and link_library variables should point to the corresponding library files in the lib directory.

```
dc_shell> echo $search_path
. /usr/synopsys/syn/I-2013.12/libraries/syn
/usr/synopsys/syn/I-2013.12/minpower/syn
/usr/synopsys/syn/I-2013.12/dw/syn_ver
/usr/synopsys/syn/I-2013.12/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>
```

3. 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.

```
dc_shell> read_verilog updown.v
Loading db file '/home/snap/lab_02/lib/saed90nm_typ.db'
Loading db file '/nfs/cad/synopsys/syn/I-2013.12/libraries/syn/gtech.db'
Loading db file '/nfs/cad/synopsys/syn/I-2013.12/libraries/syn/standard.sldb'
  Loading link library 'saed90nm_typ'
  Loading link library 'gtech'
Loading verilog file '/home/snap/lab_02/rtl/updown.v'
           ...
Presto compilation completed successfully.
Current design is now '/home/snap/lab_02/rtl/updown.db:updown'
Loaded 1 design.
Current design is 'updown'.
updown
dc_shell>
```

4. 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>
```

5. 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 designs and library components referenced in the current design and connect (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/snap/lab_02/rtl/updown.db
  saed90nm_typ (library)      /home/snap/lab_02/lib/saed90nm_typ.db
1
dc_shell>
```

6. 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:     I-2013.12
Date:        Thu Apr 10 13:13:03 2014
****************************************
                   Name                                             Total
--------------------------------------------------------------------------------
Cells                                                                2
    Cells do not drive (LINT-1)                                      2
--------------------------------------------------------------------------------
Warning: In design 'updown', cell 'B_2' does not drive any nets. (LINT-1)
Warning: In design 'updown', cell 'C35' does not drive any nets. (LINT-1)
1
dc_shell>
```

7. 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. Timing and other constraints will be discussed in detail in future lectures and modules. Similar to the previous commands, this step should return a '1' if successful.

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

8. 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_inc_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>
```

9. 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           | I-2013.12-DWBB_201312.0 |     *     |
| Licensed DW Building Blocks        |                         |           |
============================================================================
                      ...
    0:00:01     257.9      0.00       0.0       0.0
    0:00:01     257.9      0.00       0.0       0.0
    0:00:01     257.9      0.00       0.0       0.0
    0:00:01     257.9      0.00       0.0       0.0
    0:00:01     257.9      0.00       0.0       0.0
    0:00:01     257.9      0.00       0.0       0.0
    0:00:01     257.9      0.00       0.0       0.0
Loading db file '/home/snap/lab_07/lib/saed90nm_typ.db'
  Optimization Complete
  ---------------------
1
dc_shell>
```

10. 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: I-2013.12
Date   : Thu Apr 10 13:31:55 2014
****************************************
This design has no violated constraints.
1
dc_shell>
```

11. 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: I-2013.12
Date   : Thu Apr 10 13:32:59 2014
****************************************
Library(s) Used:
    saed90nm_typ (File: /home/snap/lab_07/lib/saed90nm_typ.db)
Number of ports:                          7
Number of nets:                          27
                ...
Net Interconnect area:             9.993533
Total cell area:                 247.910398
Total area:                      257.903932
1
dc_shell>
```

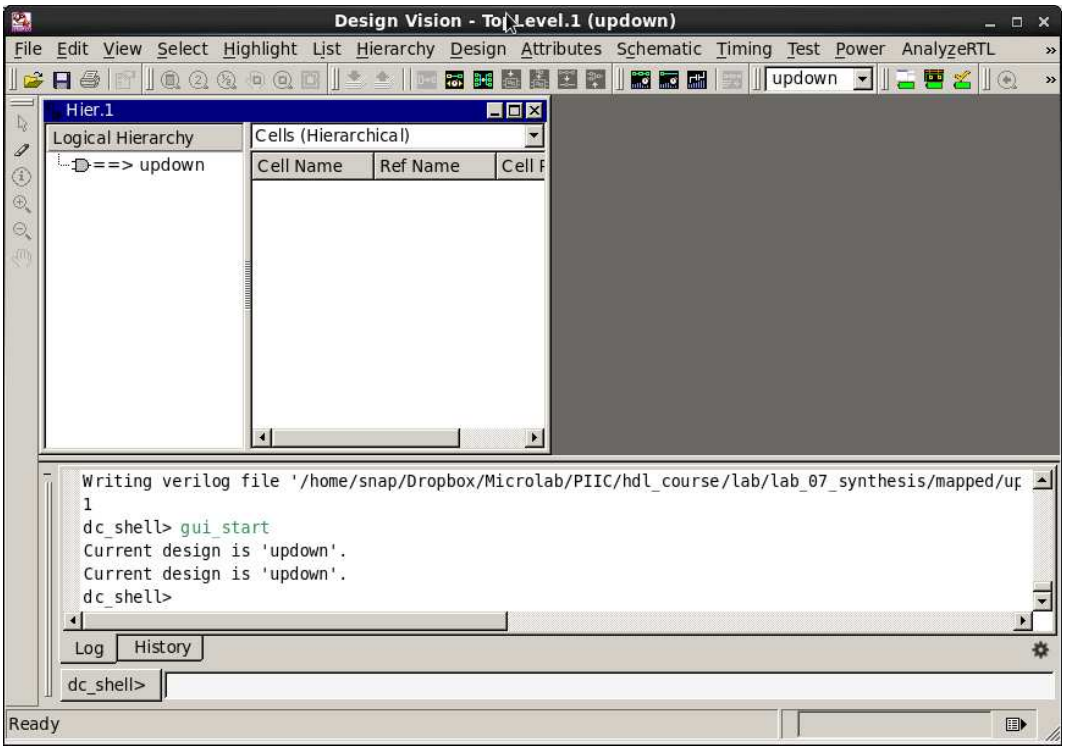
12. To view timing information of the synthesized design, run the "report_timing" command. All numerical values shown in the report have units which are defined in the libraries. In general, the report shows the amount of time allotted to the critical path (as specified in the timing constraints), the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

```
dc_shell> report_timing

****************************************
Report : timing
        -path full
        -delay max
                ...
data required time                                 8.44
  -----------------------------------------------------------
  data required time                                 8.44
  data arrival time                                  -3.35
  -----------------------------------------------------------
  slack (MET)                                         5.09
1
dc_shell>
```

13. 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/snap/lab_02/mapped/updown.sdf'. (WT-3)
1
dc_shell>
```


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

```
Writing verilog file '/home/snap/Dropbox/Microlab/PIIC/hdl_course/lab/lab_07_synthesis/mapped/up
1
dc_shell> gui_start
Current design is 'updown'.
Current design is 'updown'.
dc_shell>
```

Figure 1: Design Compiler GUI

14. 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/snap/lab_02/mapped/updown_mapped.v'.
    1
    dc_shell>
    ```

15. 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

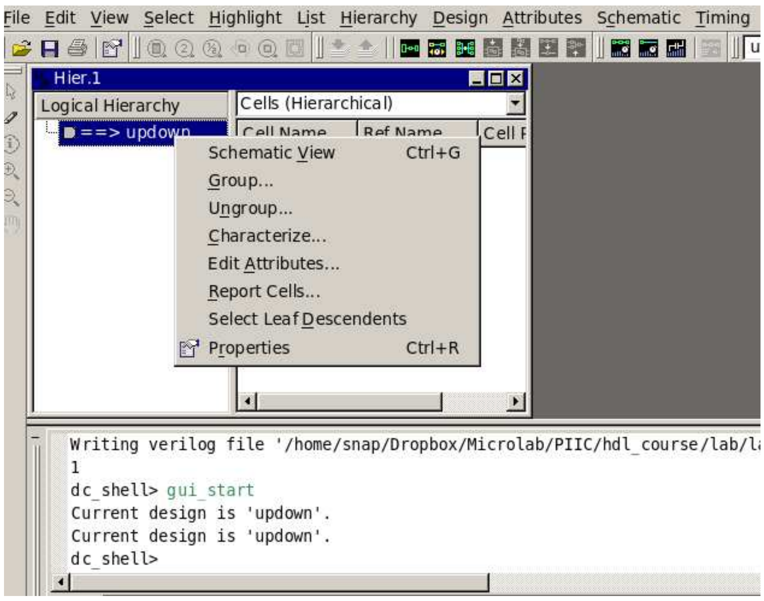
    ```
    dc_shell> gui_start
    dc_shell> Current design is 'updown'.
    Current design is 'updown'.
    dc_shell>
    ```

16. Select the updown module from the hierarchy pane and right-click. A window should appear similar to the one shown in figure 2. Select the "Schematic View" option by left-clicking on it (alternatively you can also press Ctrl+G instead of bringing up this window).

17. 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 similar window to the one shown in figure 3. Take note that you can also zoom in and out to get a better view of the schematic.

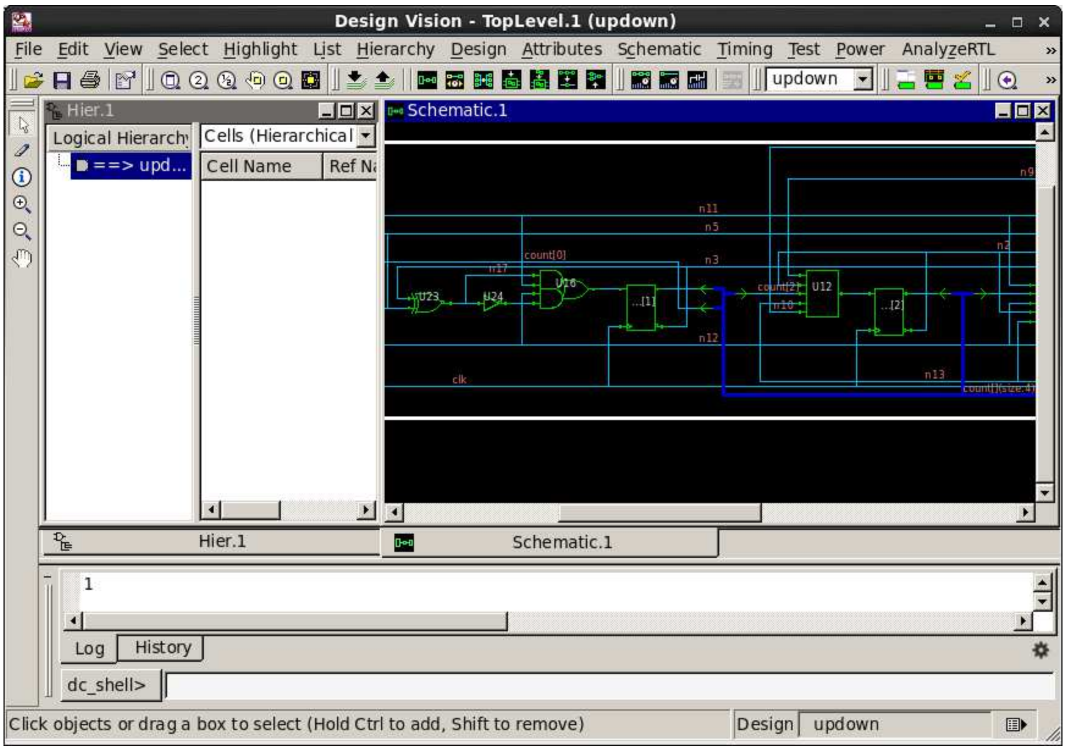
18. Exit DC by either typing quit on the command line or selecting exit on the GUI. Make sure that the synthesized design has been written out to a file by viewing the contents of the lab_02/mapped/updown_mapped.v on a text editor. You should have a file similar to the one shown below. Notice that the 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   N13, N14, N15, N16, n1, n2, n3, n5, n6, n7, n8, n9, n10, n11, n12,
             n13, n14, n15, n16, n17;
      DFFX1 \count_reg[0]  ( .D(N13), .CLK(clk), .Q(count[0]), .QN(n5) );
      DFFX1 \count_reg[1]  ( .D(N14), .CLK(clk), .Q(count[1]), .QN(n3) );
      DFFX1 \count_reg[2]  ( .D(N15), .CLK(clk), .Q(count[2]), .QN(n2) );
    ...
    ```



```
Writing verilog file '/home/snap/Dropbox/Microlab/PIIC/hdl_course/lab/l
1
dc_shell> gui_start
Current design is 'updown'.
Current design is 'updown'.
dc_shell>
```

Figure 2: Viewing the schematic



```
1
dc_shell>
```

Click objects or drag a box to select (Hold Ctrl to add, Shift to remove)    Design updown

Figure 3: Schematic view of synthesized design

## 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 lab_02/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   N13, N14, N15, N16, n1, n2, n3, n5, n6, n7, n8, n9, n10, n11, n12,
         n13, n14, n15, n16, n17;
  DFFX1 \count_reg[0]  ( .D(N13), .CLK(clk), .Q(count[0]), .QN(n5) );
  ...
```

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 use the $sdf_annotate directive. Add the $sdf_annotate line to your testbench, inside the stimulus generation part.

```
  ...
initial begin
    $vcdplusfile("tb_updown.vpd");
    $vcdpluson;
    $sdf_annotate("../mapped/updown_mapped.sdf",UUT);
    clk = 0;
    nrst = 0;
  ...
```

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 lab_02/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.

```
[snap@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 H-2013.06-SP1_Full64 -- Thu Apr 10 17:39:25 2014
               Copyright (c) 1991-2013 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 '../rtl/updown.v'
Parsing design file '../lib/saed90nm.v'
Parsing design file 'tb_updown.v'
Top Level Modules:
       AND2X2


       ....
$finish called from file "tb_updown.v", line 31.
$finish at simulation time               110000
           V C S   S i m u l a t i o n   R e p o r t
Time: 110000 ps
CPU Time:      0.500 seconds;       Data structure size:   0.3Mb
Thu Apr 10 17:39:28 2014
CPU time: 1.022 seconds to compile + .075 seconds to elab + .
240 seconds to link + .556 seconds in simulation
[snap@artanis sim]$
```

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 4.

5. Select all of the signals in the tb_updown module and display them in the waveform viewer. You should have a window similar to the one shown in figure 5. Show this simulation waveform to your instructor.

6. One important difference between synthesized and unsynthesized designs is that glitches and logic delays are already introduced in the synthesized design. Zooming in to a rising edge clock transition clearly shows this, as shown in figure 6

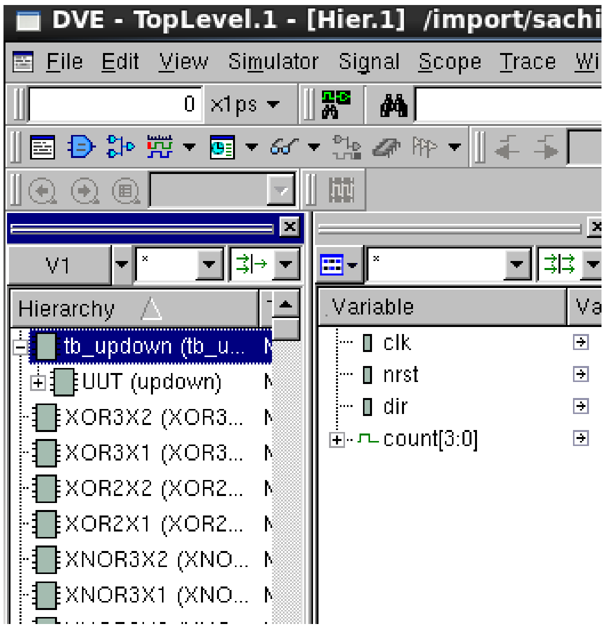


Figure 4: Testbench database hierarchy pane

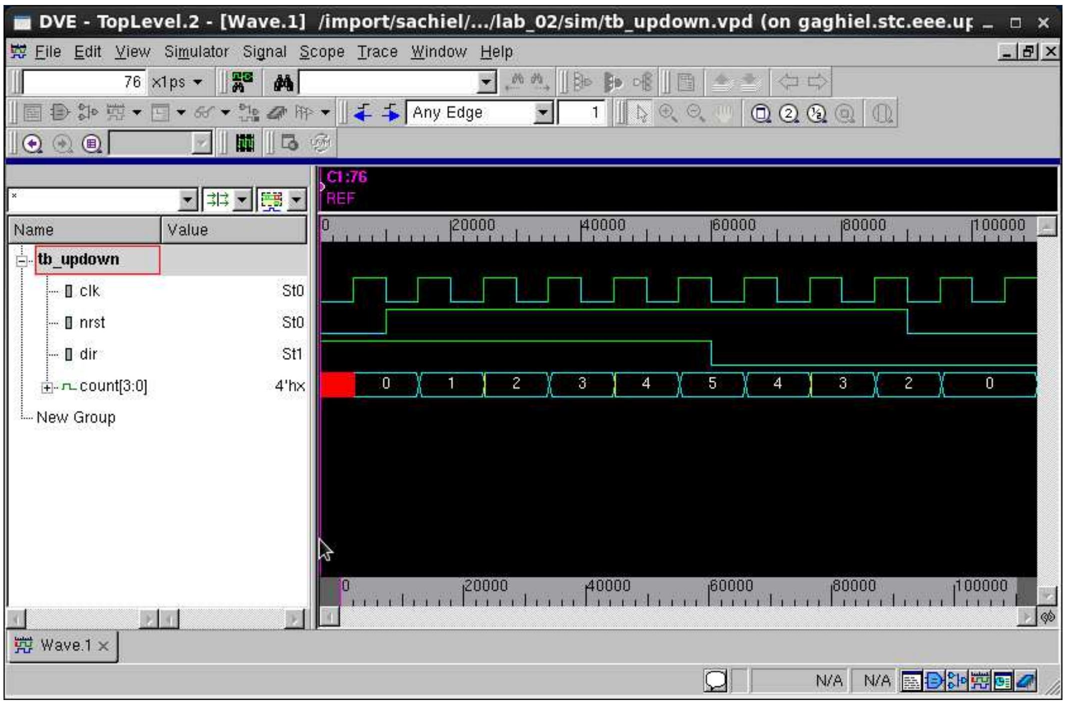


Figure 5: Testbench waveform

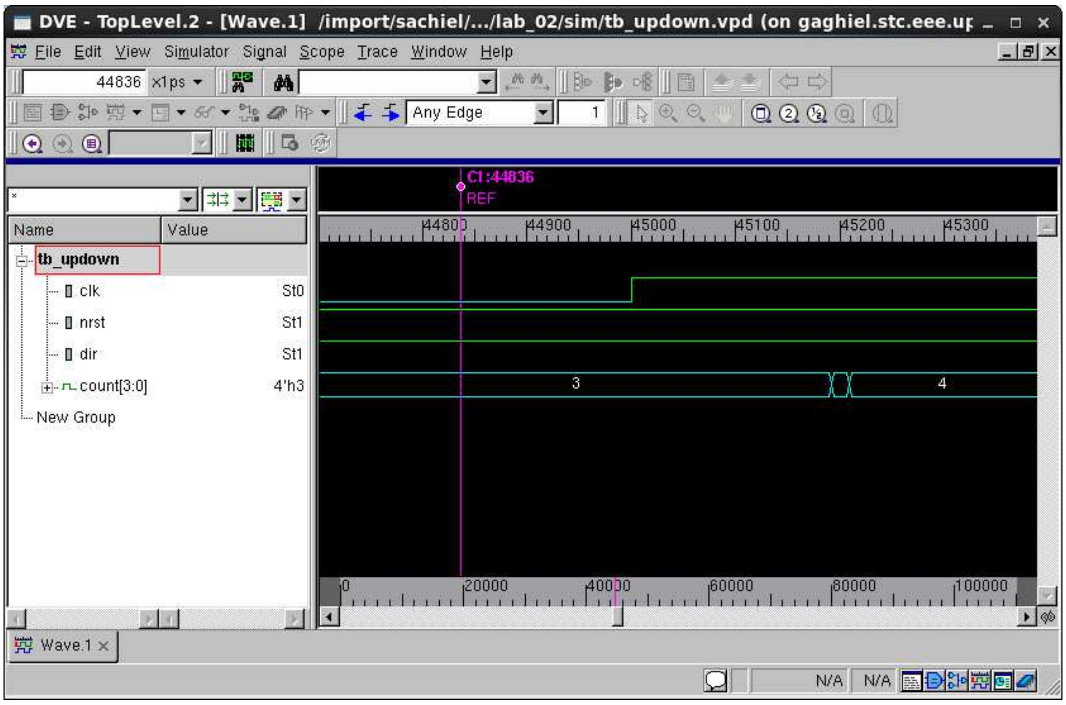


Figure 6: Glitches and logic delay