Saturday
Sep 29, 2024

EE 671: VLSI DESIGN 2024/2025 ASSIGNMENT-3

Due On
Oct 6, 2024

Note:

1. This assignment will be done in groups with one submission per group.
2. Submission is **only** through Moodle in the form of a PDF file upload.
3. All plots must be legible/readable in the submission (axes values and units).

Introduction:

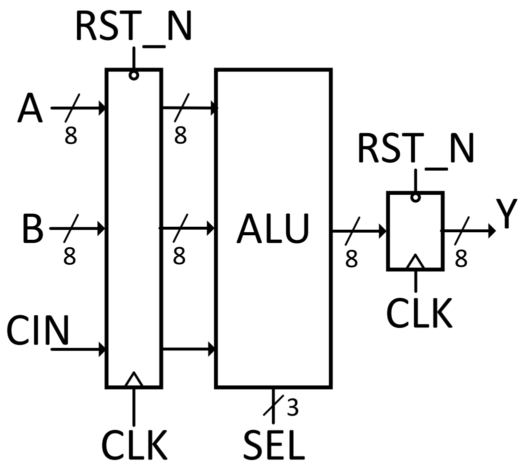
In this assignment you will perform synthesis of a design to obtain the RTL logic. This will be done through OpenLane based open-source IC design flow. The assignment is broken into two stages: (a) perform synthesis of an existing design and (b) design, verify and synthesis of a new logic. This assignment also provides some encouragement to use a unix based system and a few handy commands.

Instructions to login to the VLSI lab machine:

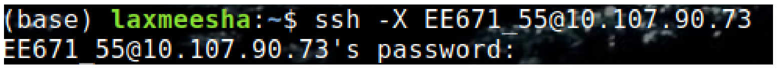
- Each group is assigned a group id for login. Refer to your login id from here. The default password is EE671
- If you are on any unix based system, open a terminal for subsequent steps. If you are on Windows, download MobaXterm and open a terminal via MobaXterm. Ensure you are connected to IITB network.
- To login, type on the terminal: *ssh -X EE671_X@10.107.90.73* (X corresponds to the number from your login id assigned to your group). Enter the password and you are good.
- After the first login, type on the terminal: *passwd* and change your password immediately.

Running synthesis on an existing example:

We consider an example ALU logic, as shown below. It has inputs A, B, CIN, SEL CLK, RST_N (active low reset) and output Y. Follow the sequence of steps below:

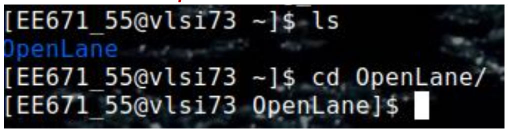


1. Login to your account



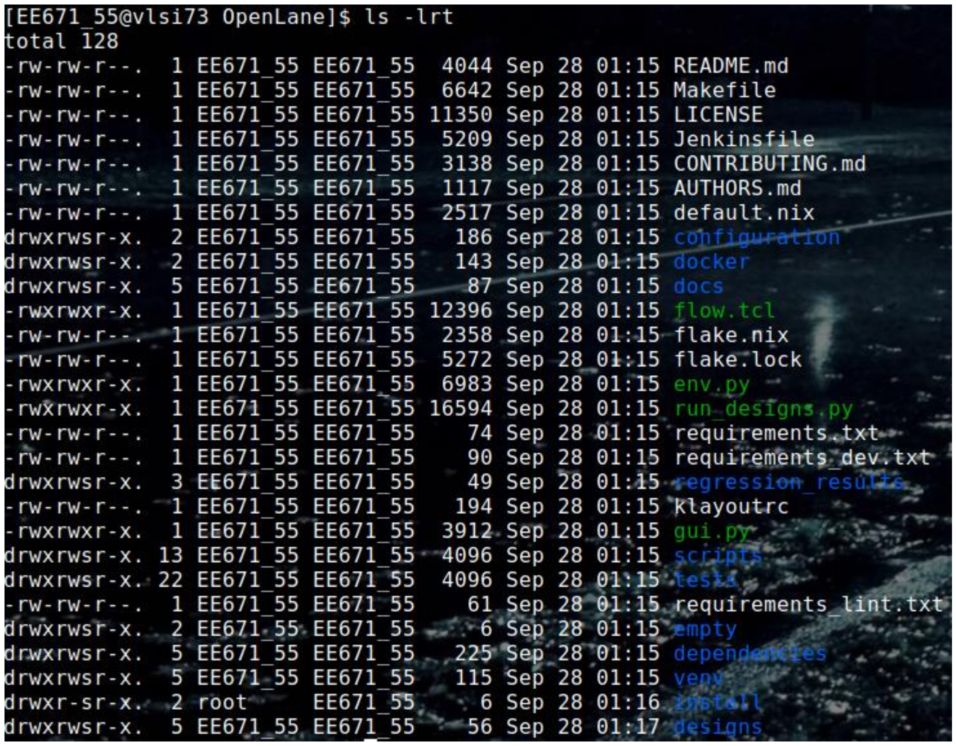
```
(base) laxmeesha:~$ ssh -X EE671_55@10.107.90.73
EE671_55@10.107.90.73's password:
```

2. Type *ls* to list the files. You should see a folder named OpenLane. Go into the directory by typing the command *cd OpenLane*



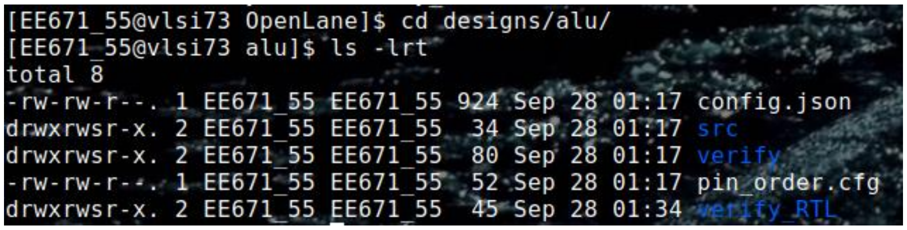
```
[EE671_55@vlsi73 ~]$ ls
OpenLane
[EE671_55@vlsi73 ~]$ cd OpenLane/
[EE671_55@vlsi73 OpenLane]$
```

3. Type *ls -lrt* to list all files with the time index (latest edited file will be on the bottom of the list). The important folder for this assignment is the "designs" folder. You are free to look inside into other files and folders. However, **DO NOT** delete/edit any files in the setup.



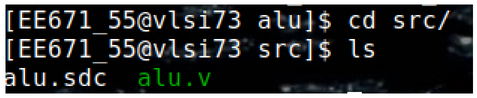
```
[EE671_55@vlsi73 OpenLane]$ ls -lrt
total 128
-rw-rw-r--.  1 EE671_55 EE671_55  4044 Sep 28 01:15 README.md
-rw-rw-r--.  1 EE671_55 EE671_55  6642 Sep 28 01:15 Makefile
-rw-rw-r--.  1 EE671_55 EE671_55 11350 Sep 28 01:15 LICENSE
-rw-rw-r--.  1 EE671_55 EE671_55  5209 Sep 28 01:15 Jenkinsfile
-rw-rw-r--.  1 EE671_55 EE671_55  3138 Sep 28 01:15 CONTRIBUTING.md
-rw-rw-r--.  1 EE671_55 EE671_55  1117 Sep 28 01:15 AUTHORS.md
-rw-rw-r--.  1 EE671_55 EE671_55  2517 Sep 28 01:15 default.nix
drwxrwsr-x.  2 EE671_55 EE671_55   186 Sep 28 01:15 configuration
drwxrwsr-x.  2 EE671_55 EE671_55   143 Sep 28 01:15 docker
drwxrwsr-x.  5 EE671_55 EE671_55    87 Sep 28 01:15 docs
-rwxrwxr-x.  1 EE671_55 EE671_55 12396 Sep 28 01:15 flow.tcl
-rw-rw-r--.  1 EE671_55 EE671_55  2358 Sep 28 01:15 flake.nix
-rw-rw-r--.  1 EE671_55 EE671_55  5272 Sep 28 01:15 flake.lock
-rwxrwxr-x.  1 EE671_55 EE671_55  6983 Sep 28 01:15 env.py
-rwxrwxr-x.  1 EE671_55 EE671_55 16594 Sep 28 01:15 run_designs.py
-rw-rw-r--.  1 EE671_55 EE671_55    74 Sep 28 01:15 requirements.txt
-rw-rw-r--.  1 EE671_55 EE671_55    90 Sep 28 01:15 requirements_dev.txt
drwxrwsr-x.  3 EE671_55 EE671_55    49 Sep 28 01:15 regression_results
-rw-rw-r--.  1 EE671_55 EE671_55   194 Sep 28 01:15 klayoutrc
-rwxrwxr-x.  1 EE671_55 EE671_55  3912 Sep 28 01:15 gui.py
drwxrwsr-x. 13 EE671_55 EE671_55  4096 Sep 28 01:15 scripts
drwxrwsr-x. 22 EE671_55 EE671_55  4096 Sep 28 01:15 tests
-rw-rw-r--.  1 EE671_55 EE671_55    61 Sep 28 01:15 requirements_lint.txt
drwxrwsr-x.  2 EE671_55 EE671_55     6 Sep 28 01:15 empty
drwxrwsr-x.  5 EE671_55 EE671_55   225 Sep 28 01:15 dependencies
drwxrwsr-x.  5 EE671_55 EE671_55   115 Sep 28 01:15 venv
drwxr-sr-x.  2 root     EE671_55     6 Sep 28 01:16 install
drwxrwsr-x.  5 EE671_55 EE671_55    56 Sep 28 01:17 designs
```

4. The Verilog code of the ALU is kept inside the designs folder. Get into the designs folder and inside that get into the alu folder and list all files. You will see three folders (a) src (b) verify (c) verify_RTL and a config.json file. The src file is where all original Verilog files are to be placed. The verify folder is used to verify the design using a test bench and verify_RTL folder is used to perform verification of the RTL.



```
[EE671_55@vlsi73 OpenLane]$ cd designs/alu/
[EE671_55@vlsi73 alu]$ ls -lrt
total 8
-rw-rw-r--. 1 EE671_55 EE671_55 924 Sep 28 01:17 config.json
drwxrwsr-x. 2 EE671_55 EE671_55  34 Sep 28 01:17 src
drwxrwsr-x. 2 EE671_55 EE671_55  80 Sep 28 01:17 verify
-rw-rw-r--. 1 EE671_55 EE671_55  52 Sep 28 01:17 pin_order.cfg
drwxrwsr-x. 2 EE671_55 EE671_55  45 Sep 28 01:34 verify_RTL
```

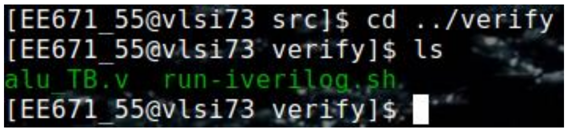
5. Get into the src folder and you should see two files alu.v and alu.sdc. We will discuss about the alu.sdc in the next assignment.



```
[EE671_55@vlsi73 alu]$ cd src/
[EE671_55@vlsi73 src]$ ls
alu.sdc  alu.v
```

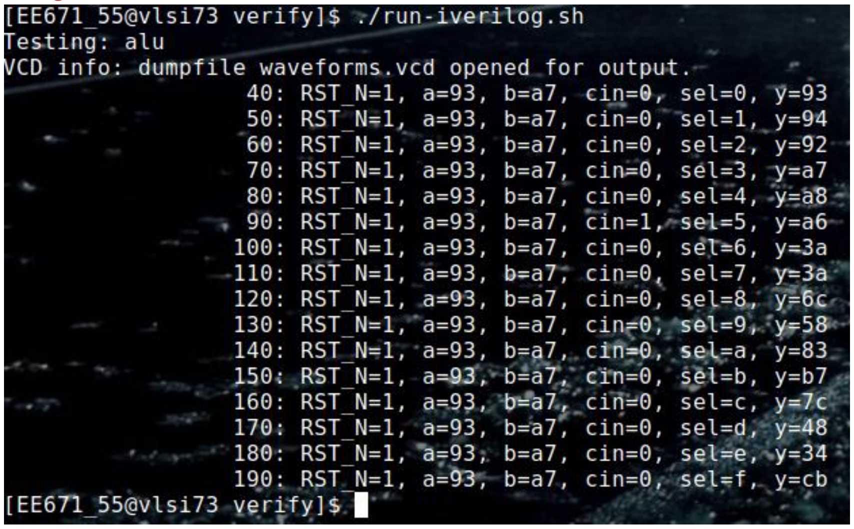
6. Open the alu.v Verilog code using any test editor and check that the design is exactly same as the figure shown above. You can use nedit or gedit text editors. For example, to open using nedit, you can type on the terminal *nedit alu.v &*

7. Now, go to the verify folder to perform a functional simulation. If you list the files under the verify folder, you will see a testbench *alu_TB.v* and a shell script to run iverilog *run-iverilog.sh*



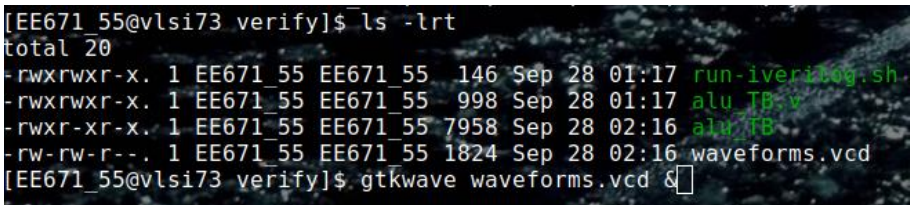
```
[EE671_55@vlsi73 src]$ cd ../verify
[EE671_55@vlsi73 verify]$ ls
alu_TB.v  run-iverilog.sh
[EE671_55@vlsi73 verify]$ 
```

8. Open both the files using nedit/gedit and check the file contents. To run iverilog, simply type *./run-iverilog.sh* on the terminal.



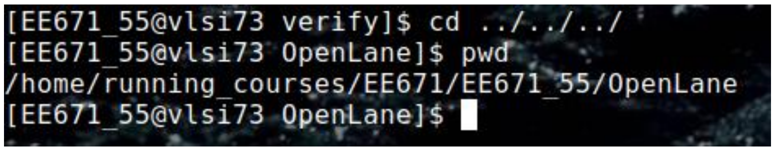
```
[EE671_55@vlsi73 verify]$ ./run-iverilog.sh
Testing: alu
VCD info: dumpfile waveforms.vcd opened for output.
                   40: RST_N=1, a=93, b=a7, cin=0, sel=0, y=93
                   50: RST_N=1, a=93, b=a7, cin=0, sel=1, y=94
                   60: RST_N=1, a=93, b=a7, cin=0, sel=2, y=92
                   70: RST_N=1, a=93, b=a7, cin=0, sel=3, y=a7
                   80: RST_N=1, a=93, b=a7, cin=0, sel=4, y=a8
                   90: RST_N=1, a=93, b=a7, cin=1, sel=5, y=a6
                  100: RST_N=1, a=93, b=a7, cin=0, sel=6, y=3a
                  110: RST_N=1, a=93, b=a7, cin=0, sel=7, y=3a
                  120: RST_N=1, a=93, b=a7, cin=0, sel=8, y=6c
                  130: RST_N=1, a=93, b=a7, cin=0, sel=9, y=58
                  140: RST_N=1, a=93, b=a7, cin=0, sel=a, y=83
                  150: RST_N=1, a=93, b=a7, cin=0, sel=b, y=b7
                  160: RST_N=1, a=93, b=a7, cin=0, sel=c, y=7c
                  170: RST_N=1, a=93, b=a7, cin=0, sel=d, y=48
                  180: RST_N=1, a=93, b=a7, cin=0, sel=e, y=34
                  190: RST_N=1, a=93, b=a7, cin=0, sel=f, y=cb
[EE671_55@vlsi73 verify]$ 
```

9. If you now list the folder contents using *ls* command, you will see a waveforms.vcd file created. We can open this file on the terminal by typing: *gtkwave waveforms.vcd &*



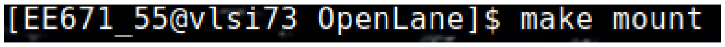
```
[EE671_55@vlsi73 verify]$ ls -lrt
total 20
-rwxrwxr-x. 1 EE671_55 EE671_55  146 Sep 28 01:17 run-iverilog.sh
-rwxrwxr-x. 1 EE671_55 EE671_55  998 Sep 28 01:17 alu_TB.v
-rwxr-xr-x. 1 EE671_55 EE671_55 7958 Sep 28 02:16 alu_TB
-rw-rw-r--. 1 EE671_55 EE671_55 1824 Sep 28 02:16 waveforms.vcd
[EE671_55@vlsi73 verify]$ gtkwave waveforms.vcd &
```

10. Check the waveforms and ensure that all the inputs and ouputs are correct.
11. Now we have a verified Verilog design which is ready for synthesis. To perform synthesis, we need to go to the OpenLane directory. cd to the OpenLane directory on the terminal. Use *pwd* command to check the "present working directory"



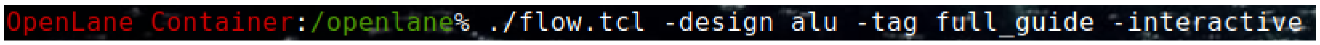
```
[EE671_55@vlsi73 verify]$ cd ../../../
[EE671_55@vlsi73 OpenLane]$ pwd
/home/running_courses/EE671/EE671_55/OpenLane
[EE671_55@vlsi73 OpenLane]$ 
```

11. We use a docker based environment to run OpenLane. To enter the docker container, type *make mount* on the terminal



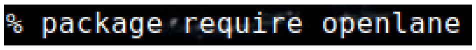
```
[EE671_55@vlsi73 OpenLane]$ make mount
```

12. You are now inside the docker container. From this container, we will run all the tcl scripts. To call the OpenLane flow, we need to run the flow.tcl script with some arguments. On the container terminal, type *./flow.tcl -design alu -tag full_guide -interactive* . The design alu points to the directory "alu" inside the designs folder that you earlier looked at. We ensured that the directory name (**alu** in this case) is same as the top cell design name and Verilog name (module **alu** in file **alu**.v).



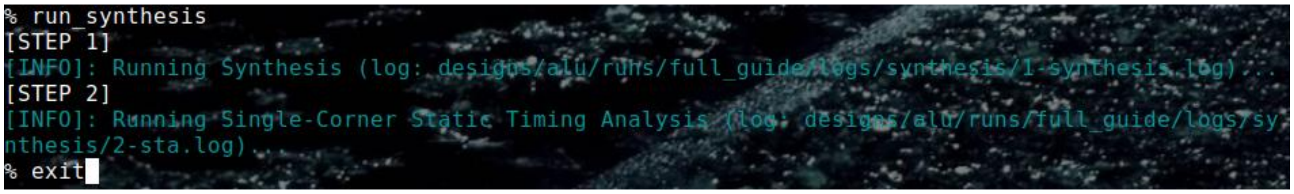
```
OpenLane Container:/openlane% ./flow.tcl -design alu -tag full_guide -interactive
```

13. Next type the following command to import OpenLane tools: *package require openlane*



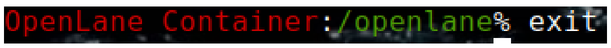
```
% package require openlane
```

14. To run synthesis, type the following: *run_synthesis* . This completes the synthesis and now exit the openlane flow by typing *exit.*



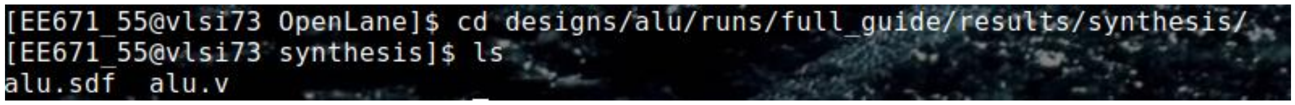
```
% run_synthesis
[STEP 1]
[INFO]: Running Synthesis (log: designs/alu/runs/full_guide/logs/synthesis/1-synthesis.log)...
[STEP 2]
[INFO]: Running Single-Corner Static Timing Analysis (log: designs/alu/runs/full_guide/logs/sy
nthesis/2-sta.log)...
% exit
```

15. Finally, exit the container by typing *exit*. You are now back on the original terminal.



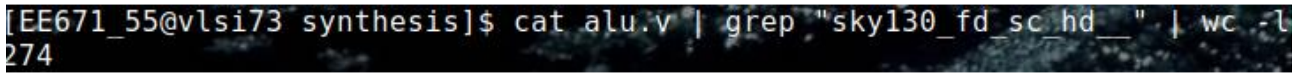
```
OpenLane Container:/openlane% exit
```

16. The RTL from synthesis is generated inside the designs folder. Cd to the *designs/alu/runs/full_guide/results/synthesis/* folder and list the files. You will see an alu.v RTL file.



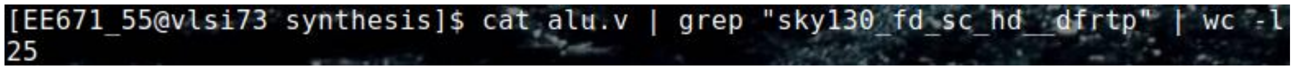
```
[EE671_55@vlsi73 OpenLane]$ cd designs/alu/runs/full_guide/results/synthesis/
[EE671_55@vlsi73 synthesis]$ ls
alu.sdf  alu.v
```

17. Open the RTL file and see that this has only standard cells. You can use nedit or gedit to open the file.
18. You can make use of unix commands to check how many standard cells are used in this RTL. You can use the cat command to read the alu.v file and pipe the output to grep command which will search for patterns. The output of grep can be piped again to a word-count/line-count command.



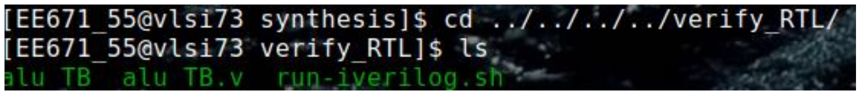
```
[EE671_55@vlsi73 synthesis]$ cat alu.v | grep "sky130_fd_sc_hd__" | wc -l
274
```

This clearly says a total of 274 cells from library "sky130_fd_sc_hd" are used in alu.v RTL. Now, if I want to know how many DFFs are used in the alu.v, only the grep pattern needs to be updated to "sky130_fd_sc_hd__dfrtp" (dfrtp is DFF with positive-edge and reset) – you will see a total of 25 DFFs.



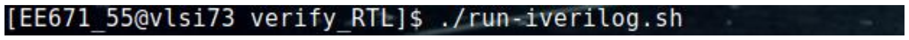
```
[EE671_55@vlsi73 synthesis]$ cat alu.v | grep "sky130_fd_sc_hd__dfrtp" | wc -l
25
```

18. Finally, we need to run the same testbench based verification on the RTL. Cd to the verify_RTL folder and you will see the alu_TB.v (this is the same TB copied from the verify folder) and a run-iverilog.sh script to run iverilog (this script is slightly modified compared to the one in the verify folder).



```
[EE671_55@vlsi73 synthesis]$ cd ../../../../verify_RTL/
[EE671_55@vlsi73 verify_RTL]$ ls
alu_TB  alu_TB.v  run-iverilog.sh
```

19. Run the script to simulate and generate the waveform file. Open using gtkwave and verify the results.



```
[EE671_55@vlsi73 verify_RTL]$ ./run-iverilog.sh
```

Q) In this assignment you will design a 16-bit Brent-Kung tree adder. The inputs A, B (16 bits each), CIN are clocked into a register using CLK signal. The registered signals are given as input to the adder and the output is again registered using a clock (both SUM and COUT are registered). Therefore, your module will have A, B, CIN, CLK, RST_N as inputs and SUM, COUT as outputs. Follow the steps below:

1. In the designs folder copy the alu folder into a new folder called **bkadder** (use the command *cp -r alu bkadder*).
2. Inside the bkadder folder, go to the src folder and rename alu.v to bkadder.v and alu.sdc to bkadder.sdc (use the command *mv alu.v bkadder.v*). Open the bkadder.v file and change the module name from alu to bkadder and put in the logic for Brent-Kung adder.
3. Once you have the code ready, go to the verify folder. Rename alu_TB.v to bkadder_TB.v. Open the testbench file using nedit and modify the TB. Please ensure you have multiple input combinations of A, B in the TB.
4. Open the run-iverilog .sh file and change the design parameter to bkadder.
5. Run the iverilog script and open the waveform using gtkwave. This completes our design verification.
6. Now go to the bkadder folder and open the config.json file using nedit. In this file, search all “alu” and replace them with “bkadder”.
7. Run the synthesis using steps mentioned above (point to the new design while running flow.tcl).
8. Once the synthesis is successful, go to the verify_RTL folder. Copy your earlier testbench from the verify folder here. Open the run-iverilog.sh file and replace alu with bkadder.
9. Run iverilog and observe the waveforms.

Your report should have the following:

1. Verilog code (with proper comments, proper indentation) pasted from the original file. You will be penalized if your indentations and comments are not maintained.
2. The Testbench file (with proper comments, proper indentation) pasted from the original file.
3. Screenshot of the gtkwave waveforms (pre-synthesis).
4. Screenshot of the run_synthesis command and the associated dump on the terminal.
5. RTL file generated from synthesis.
6. Screenshot of the gtkwave waveforms (post-synthesis).
7. The following table generated from the RTL file:

Number of combinational cells	
Number of Sequential cells (Flip flops)	
Total number of cells	

Assignment 3

Anupam Pandey (24M1197), Harsh Pujare (21d180015),
Mrunal Chourey (21d070045), Nikita Madavi (23M1160)
EE671 VLSI Design

October 6, 2024

Q1. The verilog code for implemented Brent-Kung adder is -

```
module bkadder(CLK,RST_N,A,B,CIN,SUM,COUT);
  // Input pins
  input CLK,RST_N,CIN;
  input [15:0] A,B;

  // Output pins
  output reg [15:0] SUM;
  output reg COUT;

  // Input registers
  reg [15:0] A_reg,B_reg;
  reg CIN_reg;

  // Internal signals
  wire [15:0] sum_wire;
  wire cout_internal;

    // Input registers
    always @(posedge CLK or negedge RST_N)
    begin
        if (!RST_N)
        begin
            A_reg <= 16'b0;
            B_reg <= 16'b0;
            CIN_reg <= 1'b0;
        end
        else
        begin
            A_reg <= A;
            B_reg <= B;
        CIN_reg <= CIN;
    end
end

// Brent-Kung adder logic
wire [15:0] P, G;
wire [15:0] PP, GG;

// Generate P and G
assign P = A_reg ^ B_reg;
assign G = A_reg & B_reg;

// First level
wire [7:0] P1, G1;
genvar i;
generate
    for (i = 0; i < 8; i = i + 1)
    begin : gen_level1
        assign P1[i] = P[2*i+1] & P[2*i];
        assign G1[i] = G[2*i+1] | (P[2*i+1] & G[2*i]);
    end
endgenerate

// Second level
wire [3:0] P2, G2;
generate
    for (i = 0; i < 4; i = i + 1)
    begin : gen_level2
        assign P2[i] = P1[2*i+1] & P1[2*i];
        assign G2[i] = G1[2*i+1] | (P1[2*i+1] & G1[2*i]);
    end
endgenerate

// Third level
wire [1:0] P3, G3;
assign P3[0] = P2[1] & P2[0];
assign G3[0] = G2[1] | (P2[1] & G2[0]);
assign P3[1] = P2[3] & P2[2];
assign G3[1] = G2[3] | (P2[3] & G2[2]);

// Fourth level
wire P4, G4;
assign P4 = P3[1] & P3[0];
assign G4 = G3[1] | (P3[1] & G3[0]);
```

```
   // Calculate carries
 wire [16:0] C;
 assign C[0] = CIN_reg;
 assign C[1] = G[0] | (P[0] & C[0]);
 assign C[2] = G1[0] | (P1[0] & C[0]);
 assign C[16] = G4 | (P4 & C[0]);

 generate
     for (i = 3; i < 16; i = i + 1)
     begin : gen_carries
         if (i == 4)
             assign C[i] = G2[0] | (P2[0] & C[0]);
         else if (i == 8)
             assign C[i] = G3[0] | (P3[0] & C[0]);
         else if (i % 2 == 1)
             assign C[i] = G[i-1] | (P[i-1] & C[i-1]);
         else if (i % 4 == 0)
             assign C[i] = G2[i/4-1] | (P2[i/4-1] & C[i-4]);
         else
             assign C[i] = G1[i/2-1] | (P1[i/2-1] & C[i-2]);
     end
 endgenerate

   // Calculate sum
   assign sum_wire = P ^ C[15:0];
   assign cout_wire = C[16];

   // Output registers
   always @(posedge CLK or negedge RST_N)
   begin
       if (!RST_N)
       begin
           SUM <= 16'b0;
           COUT <= 1'b0;
       end
       else
       begin
           SUM <= sum_wire;
           COUT <= cout_wire;
       end
   end

endmodule
```

Q2. The verilog code for testbench is -

```
module bkadder_tb;
  // Input pins
  reg CLK,RST_N,CIN;
  reg [15:0] A,B;

  // Output pins
  wire [15:0] SUM;
  wire COUT;

  integer i;

  // DUT Initialization
  bkadder dut(.CLK(CLK),.RST_N(RST_N),.A(A),.B(B),.CIN(CIN),.SUM(SUM),.COUT(COUT));

  always #5 CLK = ~CLK;  // Generate clock with period of 10ns

  // Initialize clock
    initial begin
        CLK = 0;
        RST_N = 0;  // Apply reset
        #20;
        RST_N = 1; // Release reset
        A = 16'h0;
        B = 16'h0;
        CIN = 1'b0;
        #20;

        // Generate random test cases
        for (i = 0; i < 10; i=i+1)
        begin
            A = $random;
            B = $random;
            CIN = $random % 2; // Random binary value (0 or 1)
            #20; // Wait for a clock edge
            $display("Test %d: A = %h, B = %h, CIN = %b, SUM = %h, COUT = %b", i, A, B,
            CIN, SUM, COUT);
        end
    end

    initial
    begin
        $dumpfile("brent_kung_adder.vcd");
        $dumpvars(0, dut);
    end
endmodule
```

Q3.

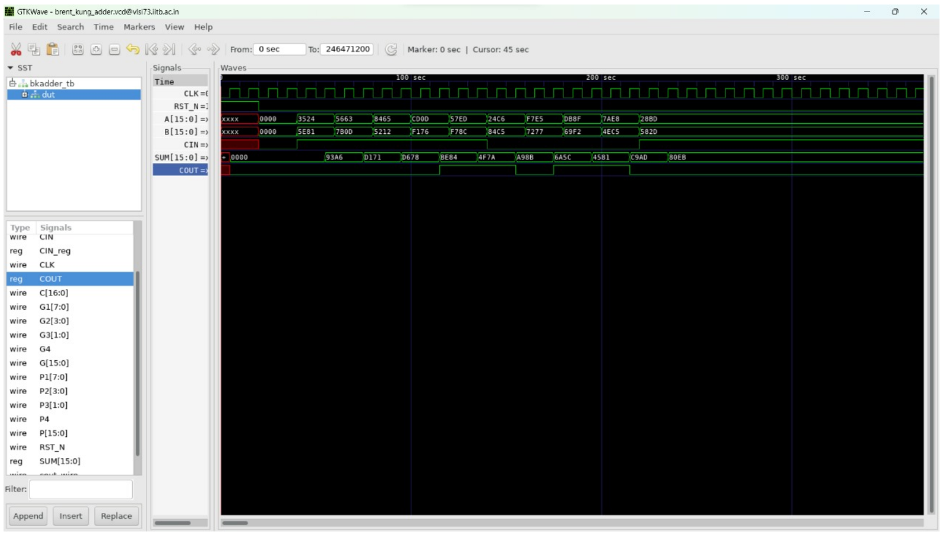
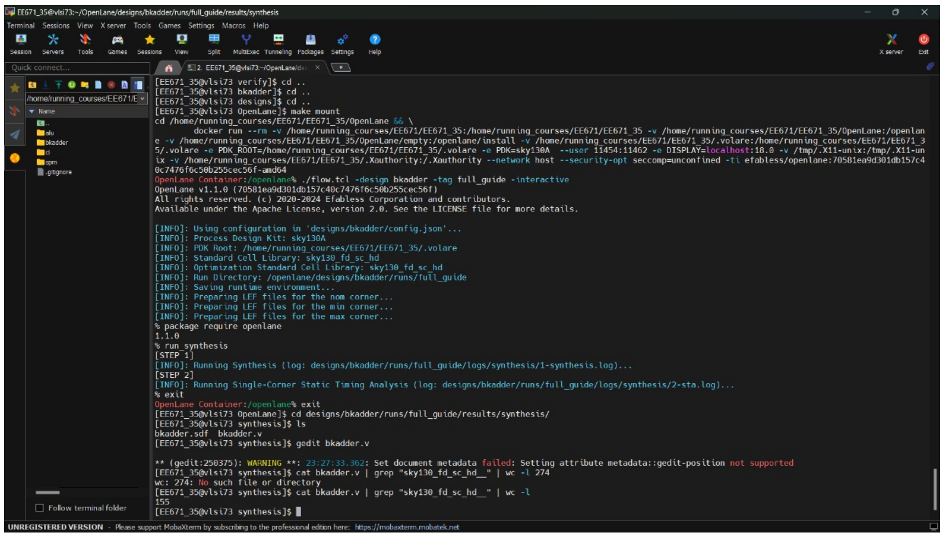


Figure 1: Pre-synthesis gtkwave waveforms

Q4.

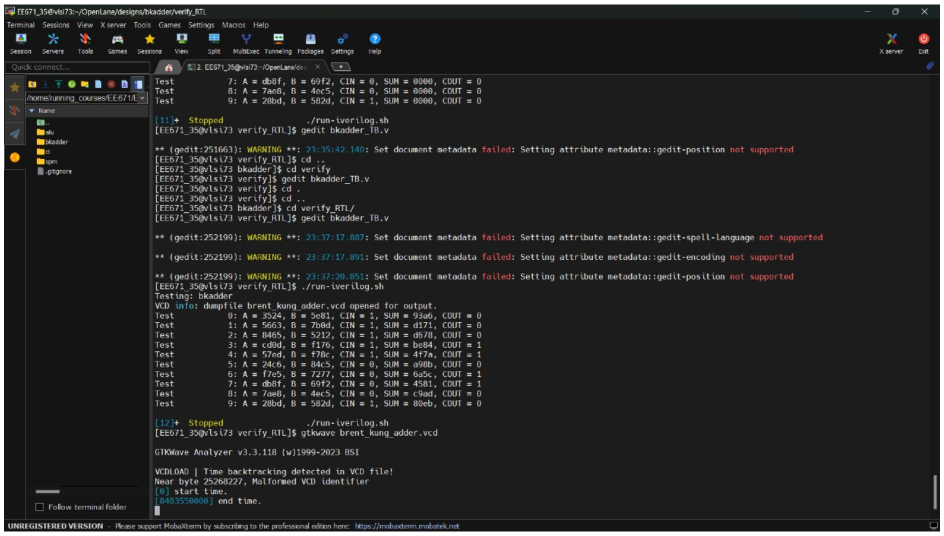

EE671_35@vlsi73:~/OpenLane/designs/bkadder/runs/full_guide/results/synthesis
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. EE671_35@vlsi73:~/OpenLane/des
/home/running_courses/EE671/E
Name
..
alu
bkadder
ci
spm
.gitignore
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```
[EE671_35@vlsi73 verify]$ cd ..
[EE671_35@vlsi73 bkadder]$ cd ..
[EE671_35@vlsi73 designs]$ cd ..
[EE671_35@vlsi73 OpenLane]$ make mount
cd /home/running_courses/EE671/EE671_35/OpenLane && \
        docker run --rm -v /home/running_courses/EE671/EE671_35:/home/running_courses/EE671/EE671_35 -v /home/running_courses/EE671/EE671_35/OpenLane:/openlan
e -v /home/running_courses/EE671/EE671_35/OpenLane/empty:/openlane/install -v /home/running_courses/EE671/EE671_35/.volare:/home/running_courses/EE671/EE671_3
5/.volare -e PDK_ROOT=/home/running_courses/EE671/EE671_35/.volare -e PDK=sky130A  --user 11454:11462 -e DISPLAY=localhost:18.0 -v /tmp/.X11-unix:/tmp/.X11-un
ix -v /home/running_courses/EE671/EE671_35/.Xauthority:/.Xauthority --network host --security-opt seccomp=unconfined -ti efabless/openlane:70581ea9d301db157c4
0c7476f6c50b255cec56f-amd64
OpenLane Container:/openlane% ./flow.tcl -design bkadder -tag full_guide -interactive
OpenLane v1.1.0 (70581ea9d301db157c40c7476f6c50b255cec56f)
All rights reserved. (c) 2020-2024 Efabless Corporation and contributors.
Available under the Apache License, version 2.0. See the LICENSE file for more details.

[INFO]: Using configuration in 'designs/bkadder/config.json'...
[INFO]: Process Design Kit: sky130A
[INFO]: PDK Root: /home/running_courses/EE671/EE671_35/.volare
[INFO]: Standard Cell Library: sky130_fd_sc_hd
[INFO]: Optimization Standard Cell Library: sky130_fd_sc_hd
[INFO]: Run Directory: /openlane/designs/bkadder/runs/full_guide
[INFO]: Saving runtime environment...
[INFO]: Preparing LEF files for the nom corner...
[INFO]: Preparing LEF files for the min corner...
[INFO]: Preparing LEF files for the max corner...
% package require openlane
1.1.0
% run_synthesis
[STEP 1]
[INFO]: Running Synthesis (log: designs/bkadder/runs/full_guide/logs/synthesis/1-synthesis.log)...
[STEP 2]
[INFO]: Running Single-Corner Static Timing Analysis (log: designs/bkadder/runs/full_guide/logs/synthesis/2-sta.log)...
% exit
OpenLane Container:/openlane% exit
[EE671_35@vlsi73 OpenLane]$ cd designs/bkadder/runs/full_guide/results/synthesis/
[EE671_35@vlsi73 synthesis]$ ls
bkadder.sdf  bkadder.v
[EE671_35@vlsi73 synthesis]$ gedit bkadder.v

** (gedit:250375): WARNING **: 23:27:33.362: Set document metadata failed: Setting attribute metadata::gedit-position not supported
[EE671_35@vlsi73 synthesis]$ cat bkadder.v | grep "sky130_fd_sc_hd__" | wc -l 274
wc: 274: No such file or directory
[EE671_35@vlsi73 synthesis]$ cat bkadder.v | grep "sky130_fd_sc_hd__" | wc -l
155
[EE671_35@vlsi73 synthesis]$ 
```

Figure 2: Screenshot of run-synthesis command



```
Test           7: A = db8f, B = 69f2, CIN = 0, SUM = 0000, COUT = 0
Test           8: A = 7ae8, B = 4ec5, CIN = 0, SUM = 0000, COUT = 0
Test           9: A = 28bd, B = 582d, CIN = 1, SUM = 0000, COUT = 0

[11]+  Stopped                 ./run-iverilog.sh
[EE671_35@vlsi73 verify_RTL]$ gedit bkadder_TB.v

** (gedit:251663): WARNING **: 23:35:42.140: Set document metadata failed: Setting attribute metadata::gedit-position not supported
[EE671_35@vlsi73 verify_RTL]$ cd ..
[EE671_35@vlsi73 bkadder]$ cd verify
[EE671_35@vlsi73 verify]$ gedit bkadder_TB.v
[EE671_35@vlsi73 verify]$ cd .
[EE671_35@vlsi73 verify]$ cd ..
[EE671_35@vlsi73 bkadder]$ cd verify_RTL/
[EE671_35@vlsi73 verify_RTL]$ gedit bkadder_TB.v

** (gedit:252199): WARNING **: 23:37:17.887: Set document metadata failed: Setting attribute metadata::gedit-spell-language not supported

** (gedit:252199): WARNING **: 23:37:17.891: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported

** (gedit:252199): WARNING **: 23:37:20.851: Set document metadata failed: Setting attribute metadata::gedit-position not supported
[EE671_35@vlsi73 verify_RTL]$ ./run-iverilog.sh
Testing: bkadder
VCD info: dumpfile brent_kung_adder.vcd opened for output.
Test           0: A = 3524, B = 5e81, CIN = 1, SUM = 93a6, COUT = 0
Test           1: A = 5663, B = 7b0d, CIN = 1, SUM = d171, COUT = 0
Test           2: A = 8465, B = 5212, CIN = 1, SUM = d678, COUT = 0
Test           3: A = cd0d, B = f176, CIN = 1, SUM = be84, COUT = 1
Test           4: A = 57ed, B = f78c, CIN = 1, SUM = 4f7a, COUT = 1
Test           5: A = 24c6, B = 84c5, CIN = 0, SUM = a98b, COUT = 0
Test           6: A = f7e5, B = 7277, CIN = 0, SUM = 6a5c, COUT = 1
Test           7: A = db8f, B = 69f2, CIN = 0, SUM = 4581, COUT = 1
Test           8: A = 7ae8, B = 4ec5, CIN = 0, SUM = c9ad, COUT = 0
Test           9: A = 28bd, B = 582d, CIN = 1, SUM = 80eb, COUT = 0

[12]+  Stopped                 ./run-iverilog.sh
[EE671_35@vlsi73 verify_RTL]$ gtkwave brent_kung_adder.vcd

GTKWave Analyzer v3.3.118 (w)1999-2023 BSI

VCDLOAD | Time backtracking detected in VCD file!
Near byte 25268227, Malformed VCD identifier
[0] start time.
[8483550000] end time.
```

Figure 3: Post-synthesis dump

Q5. RTL file generated after synthesis

```
/* Generated by Yosys 0.38 (git sha1 543faed9c8c, clang++ 17.0.6 -fPIC -Os) */

module bkadder(CLK, RST_N, A, B, CIN, SUM, COUT);
  wire _000_;
  wire _001_;
  wire _002_;
  wire _003_;
  wire _004_;
  wire _005_;
  wire _006_;
  wire _007_;
  wire _008_;
  wire _009_;
  wire _010_;
  wire _011_;
  wire _012_;
  wire _013_;
  wire _014_;
  wire _015_;
  wire _016_;
  wire _017_;
  wire _018_;
  wire _019_;
  wire _020_;
  wire _021_;
  wire _022_;
  wire _023_;
  wire _024_;
  wire _025_;
  wire _026_;
  wire _027_;
  wire _028_;
  wire _029_;
  wire _030_;
  wire _031_;
  wire _032_;
  wire _033_;
  wire _034_;
  wire _035_;
  wire _036_;
  wire _037_;
  wire _038_;
  wire _039_;
  wire _040_;
  wire _041_;
  wire _042_;
  wire _043_;
  wire _044_;
  wire _045_;
  wire _046_;
  wire _047_;
  wire _048_;
  wire _049_;
  wire _050_;
  wire _051_;
  wire _052_;
  wire _053_;
  wire _054_;
  wire _055_;
  wire _056_;
  wire _057_;
  wire _058_;
  wire _059_;
  wire _060_;
  wire _061_;
  wire _062_;
  wire _063_;
  wire _064_;
  wire _065_;
  wire _066_;
  wire _067_;
  wire _068_;
  wire _069_;
  wire _070_;
  wire _071_;
  wire _072_;
  wire _073_;
  wire _074_;
  wire _075_;
  wire _076_;
  wire _077_;
  wire _078_;
  wire _079_;
  wire _080_;
  wire _081_;
  wire _082_;
  wire _083_;
  wire _084_;
  wire _085_;
  wire _086_;
  wire _087_;
  input [15:0] A;
  wire [15:0] A;
  wire \A_reg[0] ;
  wire \A_reg[10] ;
  wire \A_reg[11] ;
  wire \A_reg[12] ;
  wire \A_reg[13] ;
  wire \A_reg[14] ;
  wire \A_reg[15] ;
  wire \A_reg[1] ;
  wire \A_reg[2] ;
  wire \A_reg[3] ;
  wire \A_reg[4] ;
  wire \A_reg[5] ;
  wire \A_reg[6] ;
  wire \A_reg[7] ;
  wire \A_reg[8] ;
  wire \A_reg[9] ;
  input [15:0] B;
  wire [15:0] B;
  wire \B_reg[0] ;
  wire \B_reg[10] ;
  wire \B_reg[11] ;
  wire \B_reg[12] ;
  wire \B_reg[13] ;
  wire \B_reg[14] ;
  wire \B_reg[15] ;
  wire \B_reg[1] ;
  wire \B_reg[2] ;
  wire \B_reg[3] ;
  wire \B_reg[4] ;
  wire \B_reg[5] ;
  wire \B_reg[6] ;
  wire \B_reg[7] ;
  wire \B_reg[8] ;
  wire \B_reg[9] ;
  input CIN;
  wire CIN;
  wire CIN_reg;
  input CLK;
  wire CLK;
  output COUT;
  wire COUT;
  input RST_N;
  wire RST_N;
  output [15:0] SUM;
  wire [15:0] SUM;
  wire cout_wire;
  wire \sum_wire[0] ;
  wire \sum_wire[10] ;
  wire \sum_wire[11] ;
  wire \sum_wire[12] ;
  wire \sum_wire[13] ;
  wire \sum_wire[14] ;
  wire \sum_wire[15] ;
  wire \sum_wire[1] ;
  wire \sum_wire[2] ;
  wire \sum_wire[3] ;
  wire \sum_wire[4] ;
  wire \sum_wire[5] ;
  wire \sum_wire[6] ;
  wire \sum_wire[7] ;
  wire \sum_wire[8] ;
  wire \sum_wire[9] ;
  sky130_fd_sc_hd__nor2_2 _088_ (
    .A(\B_reg[15] ),
    .B(\A_reg[15] ),
    .Y(_039_)
  );
  sky130_fd_sc_hd__nand2_2 _089_ (
    .A(\B_reg[14] ),
    .B(\A_reg[14] ),
    .Y(_040_)
  );
  sky130_fd_sc_hd__or2_2 _090_ (
    .A(\B_reg[14] ),
    .B(\A_reg[14] ),
    .X(_041_)
  );
  sky130_fd_sc_hd__nand2_2 _091_ (
    .A(_040_),
    .B(_041_),
    .Y(_042_)
  );
  sky130_fd_sc_hd__nand2_2 _092_ (
    .A(\B_reg[12] ),
    .B(\A_reg[12] ),
    .Y(_043_)
  );
  sky130_fd_sc_hd__nor2_2 _093_ (
    .A(\B_reg[7] ),
    .B(\A_reg[7] ),
    .Y(_044_)
  );
  sky130_fd_sc_hd__and2_2 _094_ (
    .A(\B_reg[7] ),
    .B(\A_reg[7] ),
    .X(_045_)
  );
  sky130_fd_sc_hd__nand2_2 _095_ (
    .A(\B_reg[6] ),
    .B(\A_reg[6] ),
    .Y(_046_)
  );
  sky130_fd_sc_hd__or2_2 _096_ (
    .A(\B_reg[6] ),
    .B(\A_reg[6] ),
    .X(_047_)
  );
  sky130_fd_sc_hd__nand2_2 _097_ (
    .A(_046_),
    .B(_047_),
    .Y(_048_)
  );
  sky130_fd_sc_hd__o21ba_2 _098_ (
    .A1(_044_),
    .A2(_046_),
    .B1_N(_045_),
    .X(_049_)
  );
  sky130_fd_sc_hd__o31a_2 _099_ (
    .A1(_044_),
    .A2(_045_),
    .A3(_048_),
    .B1(_049_),
    .X(_050_)
  );
  sky130_fd_sc_hd__nand2_2 _100_ (
    .A(\B_reg[11] ),
    .B(\A_reg[11] ),
    .Y(_051_)
  );
  sky130_fd_sc_hd__or2_2 _101_ (
    .A(\B_reg[11] ),
    .B(\A_reg[11] ),
    .X(_052_)
  );
  sky130_fd_sc_hd__nand2_2 _102_ (
    .A(_051_),
    .B(_052_),
    .Y(_053_)
  );
  sky130_fd_sc_hd__and2_2 _103_ (
    .A(\B_reg[10] ),
    .B(\A_reg[10] ),
    .X(_054_)
  );
  sky130_fd_sc_hd__nor2_2 _104_ (
    .A(\B_reg[10] ),
    .B(\A_reg[10] ),
    .Y(_055_)
  );
  sky130_fd_sc_hd__nor2_2 _105_ (
    .A(_054_),
    .B(_055_),
    .Y(_056_)
  );
  sky130_fd_sc_hd__or2b_2 _106_ (
    .A(_053_),
    .B_N(_056_),
    .X(_057_)
```

```
);
sky130_fd_sc_hd__nand2_2 _107_ (
  .A(\B_reg[8] ),
  .B(\A_reg[8] ),
  .Y(_058_)
);
sky130_fd_sc_hd__or2_2 _108_ (
  .A(\B_reg[8] ),
  .B(\A_reg[8] ),
  .X(_059_)
);
sky130_fd_sc_hd__nand2_2 _109_ (
  .A(_058_),
  .B(_059_),
  .Y(_060_)
);
sky130_fd_sc_hd__nand2_2 _110_ (
  .A(\B_reg[9] ),
  .B(\A_reg[9] ),
  .Y(_061_)
);
sky130_fd_sc_hd__or2_2 _111_ (
  .A(\B_reg[9] ),
  .B(\A_reg[9] ),
  .X(_062_)
);
sky130_fd_sc_hd__nand2_2 _112_ (
  .A(_061_),
  .B(_062_),
  .Y(_063_)
);
sky130_fd_sc_hd__or2_2 _113_ (
  .A(_060_),
  .B(_063_),
  .X(_064_)
);
sky130_fd_sc_hd__nand2_2 _114_ (
  .A(\B_reg[5] ),
  .B(\A_reg[5] ),
  .Y(_065_)
);
sky130_fd_sc_hd__or2_2 _115_ (
  .A(\B_reg[5] ),
  .B(\A_reg[5] ),
  .X(_066_)
```

```
);
sky130_fd_sc_hd__nand2_2 _116_ (
  .A(_065_),
  .B(_066_),
  .Y(_067_)
);
sky130_fd_sc_hd__inv_2 _117_ (
  .A(_067_),
  .Y(_068_)
);
sky130_fd_sc_hd__nand2_2 _118_ (
  .A(\B_reg[4] ),
  .B(\A_reg[4] ),
  .Y(_069_)
);
sky130_fd_sc_hd__or2_2 _119_ (
  .A(\B_reg[4] ),
  .B(\A_reg[4] ),
  .X(_070_)
);
sky130_fd_sc_hd__and2_2 _120_ (
  .A(_069_),
  .B(_070_),
  .X(_071_)
);
sky130_fd_sc_hd__or2_2 _121_ (
  .A(\B_reg[3] ),
  .B(\A_reg[3] ),
  .X(_072_)
);
sky130_fd_sc_hd__xor2_2 _122_ (
  .A(\B_reg[2] ),
  .B(\A_reg[2] ),
  .X(_073_)
);
sky130_fd_sc_hd__or2_2 _123_ (
  .A(\B_reg[1] ),
  .B(\A_reg[1] ),
  .X(_074_)
);
sky130_fd_sc_hd__a22o_2 _124_ (
  .A1(\B_reg[1] ),
  .A2(\A_reg[1] ),
  .B1(\B_reg[0] ),
  .B2(\A_reg[0] ),
  .X(_075_)
);
sky130_fd_sc_hd__a22o_2 _125_ (
  .A1(\B_reg[3] ),
  .A2(\A_reg[3] ),
  .B1(\B_reg[2] ),
  .B2(\A_reg[2] ),
  .X(_076_)
);
sky130_fd_sc_hd__a31o_2 _126_ (
  .A1(_073_),
  .A2(_074_),
  .A3(_075_),
  .B1(_076_),
  .X(_077_)
);
sky130_fd_sc_hd__inv_2 _127_ (
  .A(CIN_reg),
  .Y(_078_)
);
sky130_fd_sc_hd__xnor2_2 _128_ (
  .A(\B_reg[0] ),
  .B(\A_reg[0] ),
  .Y(_079_)
);
sky130_fd_sc_hd__nand2_2 _129_ (
  .A(\B_reg[1] ),
  .B(\A_reg[1] ),
  .Y(_080_)
);
sky130_fd_sc_hd__and4bb_2 _130_ (
  .A_N(_078_),
  .B_N(_079_),
  .C(_080_),
  .D(_074_),
  .X(_081_)
);
sky130_fd_sc_hd__nand2_2 _131_ (
  .A(\B_reg[3] ),
  .B(\A_reg[3] ),
  .Y(_082_)
);
sky130_fd_sc_hd__and3_2 _132_ (
  .A(_072_),
  .B(_073_),
  .C(_082_),
  .X(_083_)
);
sky130_fd_sc_hd__a22o_2 _133_ (
  .A1(_072_),
  .A2(_077_),
  .B1(_081_),
  .B2(_083_),
  .X(_084_)
);
sky130_fd_sc_hd__a21bo_2 _134_ (
  .A1(_065_),
  .A2(_069_),
  .B1_N(_066_),
  .X(_085_)
);
sky130_fd_sc_hd__nand2_2 _135_ (
  .A(_085_),
  .B(_049_),
  .Y(_086_)
);
sky130_fd_sc_hd__a31o_2 _136_ (
  .A1(_068_),
  .A2(_071_),
  .A3(_084_),
  .B1(_086_),
  .X(_087_)
);
sky130_fd_sc_hd__or4b_2 _137_ (
  .A(_050_),
  .B(_057_),
  .C(_064_),
  .D_N(_087_),
  .X(_000_)
);
sky130_fd_sc_hd__nor2_2 _138_ (
  .A(\B_reg[11] ),
  .B(\A_reg[11] ),
  .Y(_001_)
);
sky130_fd_sc_hd__nand2_2 _139_ (
  .A(\B_reg[10] ),
  .B(\A_reg[10] ),
  .Y(_002_)
);
```

```
sky130_fd_sc_hd__inv_2 _140_ (
  .A(_062_),
  .Y(_003_)
);
sky130_fd_sc_hd__a211o_2 _141_ (
  .A1(_058_),
  .A2(_061_),
  .B1(_003_),
  .C1(_057_),
  .X(_004_)
);
sky130_fd_sc_hd__o211a_2 _142_ (
  .A1(_001_),
  .A2(_002_),
  .B1(_004_),
  .C1(_051_),
  .X(_005_)
);
sky130_fd_sc_hd__or2_2 _143_ (
  .A(\B_reg[12] ),
  .B(\A_reg[12] ),
  .X(_006_)
);
sky130_fd_sc_hd__and2_2 _144_ (
  .A(_043_),
  .B(_006_),
  .X(_007_)
);
sky130_fd_sc_hd__inv_2 _145_ (
  .A(_007_),
  .Y(_008_)
);
sky130_fd_sc_hd__a21o_2 _146_ (
  .A1(_000_),
  .A2(_005_),
  .B1(_008_),
  .X(_009_)
);
sky130_fd_sc_hd__nand2_2 _147_ (
  .A(\B_reg[13] ),
  .B(\A_reg[13] ),
  .Y(_010_)
);
sky130_fd_sc_hd__nor2_2 _148_ (
  .A(\B_reg[13] ),
  .B(\A_reg[13] ),
  .Y(_011_)
);
sky130_fd_sc_hd__a31o_2 _149_ (
  .A1(_043_),
  .A2(_009_),
  .A3(_010_),
  .B1(_011_),
  .X(_012_)
);
sky130_fd_sc_hd__o21a_2 _150_ (
  .A1(_042_),
  .A2(_012_),
  .B1(_040_),
  .X(_013_)
);
sky130_fd_sc_hd__and2_2 _151_ (
  .A(\B_reg[15] ),
  .B(\A_reg[15] ),
  .X(_014_)
);
sky130_fd_sc_hd__o21bai_2 _152_ (
  .A1(_039_),
  .A2(_013_),
  .B1_N(_014_),
  .Y(cout_wire)
);
sky130_fd_sc_hd__xnor2_2 _153_ (
  .A(CIN_reg),
  .B(_079_),
  .Y(\sum_wire[0] )
);
sky130_fd_sc_hd__nand2_2 _154_ (
  .A(_074_),
  .B(_080_),
  .Y(_015_)
);
sky130_fd_sc_hd__nand2_2 _155_ (
  .A(\B_reg[0] ),
  .B(\A_reg[0] ),
  .Y(_016_)
);
sky130_fd_sc_hd__o21ai_2 _156_ (
  .A1(_078_),
  .A2(_079_),
  .B1(_016_),
  .Y(_017_)
);
sky130_fd_sc_hd__xnor2_2 _157_ (
  .A(_015_),
  .B(_017_),
  .Y(\sum_wire[1] )
);
sky130_fd_sc_hd__a21o_2 _158_ (
  .A1(_074_),
  .A2(_075_),
  .B1(_081_),
  .X(_018_)
);
sky130_fd_sc_hd__xor2_2 _159_ (
  .A(_073_),
  .B(_018_),
  .X(\sum_wire[2] )
);
sky130_fd_sc_hd__nand2_2 _160_ (
  .A(_072_),
  .B(_082_),
  .Y(_019_)
);
sky130_fd_sc_hd__a22o_2 _161_ (
  .A1(\B_reg[2] ),
  .A2(\A_reg[2] ),
  .B1(_073_),
  .B2(_018_),
  .X(_020_)
);
sky130_fd_sc_hd__xnor2_2 _162_ (
  .A(_019_),
  .B(_020_),
  .Y(\sum_wire[3] )
);
sky130_fd_sc_hd__nand2_2 _163_ (
  .A(_071_),
  .B(_084_),
  .Y(_021_)
);
sky130_fd_sc_hd__or2_2 _164_ (
  .A(_071_),
  .B(_084_),
  .X(_022_)
```

```
);
sky130_fd_sc_hd__and2_2 _165_ (
  .A(_021_),
  .B(_022_),
  .X(_023_)
);
sky130_fd_sc_hd__buf_1 _166_ (
  .A(_023_),
  .X(\sum_wire[4] )
);
sky130_fd_sc_hd__and3_2 _167_ (
  .A(_067_),
  .B(_069_),
  .C(_021_),
  .X(_024_)
);
sky130_fd_sc_hd__a21oi_2 _168_ (
  .A1(_069_),
  .A2(_021_),
  .B1(_067_),
  .Y(_025_)
);
sky130_fd_sc_hd__nor2_2 _169_ (
  .A(_024_),
  .B(_025_),
  .Y(\sum_wire[5] )
);
sky130_fd_sc_hd__o21a_2 _170_ (
  .A1(_067_),
  .A2(_021_),
  .B1(_085_),
  .X(_026_)
);
sky130_fd_sc_hd__xor2_2 _171_ (
  .A(_048_),
  .B(_026_),
  .X(\sum_wire[6] )
);
sky130_fd_sc_hd__nor2_2 _172_ (
  .A(_044_),
  .B(_045_),
  .Y(_027_)
);
sky130_fd_sc_hd__o21a_2 _173_ (
  .A1(_048_),
  .A2(_026_),
  .B1(_046_),
  .X(_028_)
);
sky130_fd_sc_hd__xnor2_2 _174_ (
  .A(_027_),
  .B(_028_),
  .Y(\sum_wire[7] )
);
sky130_fd_sc_hd__or2b_2 _175_ (
  .A(_050_),
  .B_N(_087_),
  .X(_029_)
);
sky130_fd_sc_hd__xor2_2 _176_ (
  .A(_029_),
  .B(_060_),
  .X(\sum_wire[8] )
);
sky130_fd_sc_hd__or2_2 _177_ (
  .A(_029_),
  .B(_060_),
  .X(_030_)
);
sky130_fd_sc_hd__nand2_2 _178_ (
  .A(_058_),
  .B(_030_),
  .Y(_031_)
);
sky130_fd_sc_hd__xnor2_2 _179_ (
  .A(_063_),
  .B(_031_),
  .Y(\sum_wire[9] )
);
sky130_fd_sc_hd__a31o_2 _180_ (
  .A1(_058_),
  .A2(_030_),
  .A3(_061_),
  .B1(_003_),
  .X(_032_)
);
sky130_fd_sc_hd__xnor2_2 _181_ (
  .A(_056_),
  .B(_032_),
  .Y(\sum_wire[10] )
```

```
);
sky130_fd_sc_hd__o21ai_2 _182_ (
  .A1(_055_),
  .A2(_032_),
  .B1(_002_),
  .Y(_033_)
);
sky130_fd_sc_hd__xnor2_2 _183_ (
  .A(_053_),
  .B(_033_),
  .Y(\sum_wire[11] )
);
sky130_fd_sc_hd__a21oi_2 _184_ (
  .A1(_000_),
  .A2(_005_),
  .B1(_008_),
  .Y(_034_)
);
sky130_fd_sc_hd__and3_2 _185_ (
  .A(_008_),
  .B(_000_),
  .C(_005_),
  .X(_035_)
);
sky130_fd_sc_hd__nor2_2 _186_ (
  .A(_034_),
  .B(_035_),
  .Y(\sum_wire[12] )
);
sky130_fd_sc_hd__or2b_2 _187_ (
  .A(_011_),
  .B_N(_010_),
  .X(_036_)
);
sky130_fd_sc_hd__nand2_2 _188_ (
  .A(_043_),
  .B(_009_),
  .Y(_037_)
);
sky130_fd_sc_hd__xnor2_2 _189_ (
  .A(_036_),
  .B(_037_),
  .Y(\sum_wire[13] )
);
sky130_fd_sc_hd__xor2_2 _190_ (
  .A(_042_),
  .B(_012_),
  .X(\sum_wire[14] )
);
sky130_fd_sc_hd__nor2_2 _191_ (
  .A(_014_),
  .B(_039_),
  .Y(_038_)
);
sky130_fd_sc_hd__xnor2_2 _192_ (
  .A(_038_),
  .B(_013_),
  .Y(\sum_wire[15] )
);
sky130_fd_sc_hd__dfrtp_2 _193_ (
  .CLK(CLK),
  .D(cout_wire),
  .Q(COUT),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _194_ (
  .CLK(CLK),
  .D(\sum_wire[0] ),
  .Q(SUM[0]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _195_ (
  .CLK(CLK),
  .D(\sum_wire[1] ),
  .Q(SUM[1]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _196_ (
  .CLK(CLK),
  .D(\sum_wire[2] ),
  .Q(SUM[2]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _197_ (
  .CLK(CLK),
  .D(\sum_wire[3] ),
  .Q(SUM[3]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _198_ (
  .CLK(CLK),
  .D(\sum_wire[4] ),
  .Q(SUM[4]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _199_ (
  .CLK(CLK),
  .D(\sum_wire[5] ),
  .Q(SUM[5]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _200_ (
  .CLK(CLK),
  .D(\sum_wire[6] ),
  .Q(SUM[6]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _201_ (
  .CLK(CLK),
  .D(\sum_wire[7] ),
  .Q(SUM[7]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _202_ (
  .CLK(CLK),
  .D(\sum_wire[8] ),
  .Q(SUM[8]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _203_ (
  .CLK(CLK),
  .D(\sum_wire[9] ),
  .Q(SUM[9]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _204_ (
  .CLK(CLK),
  .D(\sum_wire[10] ),
  .Q(SUM[10]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _205_ (
  .CLK(CLK),
  .D(\sum_wire[11] ),
  .Q(SUM[11]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _206_ (
  .CLK(CLK),
  .D(\sum_wire[12] ),
  .Q(SUM[12]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _207_ (
  .CLK(CLK),
  .D(\sum_wire[13] ),
  .Q(SUM[13]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _208_ (
  .CLK(CLK),
  .D(\sum_wire[14] ),
  .Q(SUM[14]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _209_ (
  .CLK(CLK),
  .D(\sum_wire[15] ),
  .Q(SUM[15]),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _210_ (
  .CLK(CLK),
  .D(A[0]),
  .Q(\A_reg[0] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _211_ (
  .CLK(CLK),
  .D(A[1]),
  .Q(\A_reg[1] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _212_ (
  .CLK(CLK),
  .D(A[2]),
  .Q(\A_reg[2] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _213_ (
  .CLK(CLK),
  .D(A[3]),
  .Q(\A_reg[3] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _214_ (
  .CLK(CLK),
  .D(A[4]),
  .Q(\A_reg[4] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _215_ (
  .CLK(CLK),
  .D(A[5]),
  .Q(\A_reg[5] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _216_ (
  .CLK(CLK),
  .D(A[6]),
  .Q(\A_reg[6] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _217_ (
  .CLK(CLK),
  .D(A[7]),
  .Q(\A_reg[7] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _218_ (
  .CLK(CLK),
  .D(A[8]),
  .Q(\A_reg[8] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _219_ (
  .CLK(CLK),
  .D(A[9]),
  .Q(\A_reg[9] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _220_ (
  .CLK(CLK),
  .D(A[10]),
  .Q(\A_reg[10] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _221_ (
  .CLK(CLK),
  .D(A[11]),
  .Q(\A_reg[11] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _222_ (
  .CLK(CLK),
  .D(A[12]),
  .Q(\A_reg[12] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _223_ (
  .CLK(CLK),
  .D(A[13]),
  .Q(\A_reg[13] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _224_ (
  .CLK(CLK),
  .D(A[14]),
  .Q(\A_reg[14] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _225_ (
  .CLK(CLK),
  .D(A[15]),
  .Q(\A_reg[15] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _226_ (
  .CLK(CLK),
  .D(B[0]),
  .Q(\B_reg[0] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _227_ (
  .CLK(CLK),
  .D(B[1]),
  .Q(\B_reg[1] ),
  .RESET_B(RST_N)
);
sky130_fd_sc_hd__dfrtp_2 _228_ (
    .CLK(CLK),
    .D(B[2]),
    .Q(\B_reg[2] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _229_ (
    .CLK(CLK),
    .D(B[3]),
    .Q(\B_reg[3] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _230_ (
    .CLK(CLK),
    .D(B[4]),
    .Q(\B_reg[4] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _231_ (
    .CLK(CLK),
    .D(B[5]),
    .Q(\B_reg[5] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _232_ (
    .CLK(CLK),
    .D(B[6]),
    .Q(\B_reg[6] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _233_ (
    .CLK(CLK),
    .D(B[7]),
    .Q(\B_reg[7] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _234_ (
    .CLK(CLK),
    .D(B[8]),
    .Q(\B_reg[8] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _235_ (
    .CLK(CLK),
    .D(B[9]),
    .Q(\B_reg[9] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _236_ (
    .CLK(CLK),
    .D(B[10]),
    .Q(\B_reg[10] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _237_ (
    .CLK(CLK),
    .D(B[11]),
    .Q(\B_reg[11] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _238_ (
    .CLK(CLK),
    .D(B[12]),
    .Q(\B_reg[12] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _239_ (
    .CLK(CLK),
    .D(B[13]),
    .Q(\B_reg[13] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _240_ (
    .CLK(CLK),
    .D(B[14]),
    .Q(\B_reg[14] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _241_ (
    .CLK(CLK),
    .D(B[15]),
    .Q(\B_reg[15] ),
    .RESET_B(RST_N)
  );
  sky130_fd_sc_hd__dfrtp_2 _242_ (
    .CLK(CLK),
    .D(CIN),
    .Q(CIN_reg),
    .RESET_B(RST_N)
  );
endmodule
```

Q6.

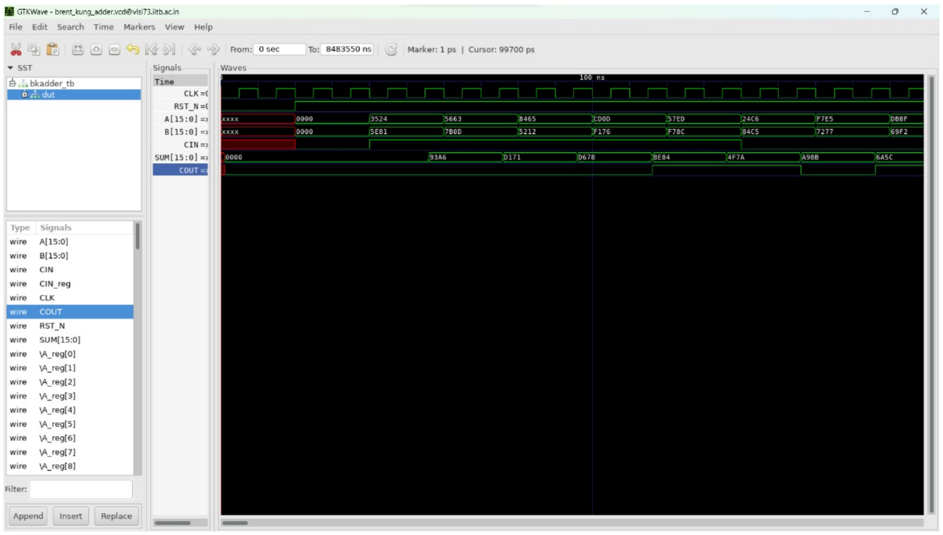


Figure 4: Post-synthesis gtkwave waveforms

Q7.

Number of combinational cells	105
Number of sequential cells	50
Total number of cells (ps)	155

Table 1: Dynamic Characteristics