EESM5020 Digital VLSI System Design and Design Automation

Design Assignment 1: ASIC Design Flow

BaohuiXie

Objective

In this assignment, you need to go through the complete ASIC design flow for a 32-bit **Brent-Kung** adder. Before starting the design, please refer to the sample project of an N-bit divider provided in the tutorial on Canvas. You can use this sample project as a template for designing the **Brent-Kung** adder. Besides, please carefully read the instructions in "Tutorial_EESM5020_Spring_2022.pdf" and "Verilog_VLSI_Tutorial.pdf" for using the design tools. After the assignment, you will gain essential understandings about ASIC design flow.

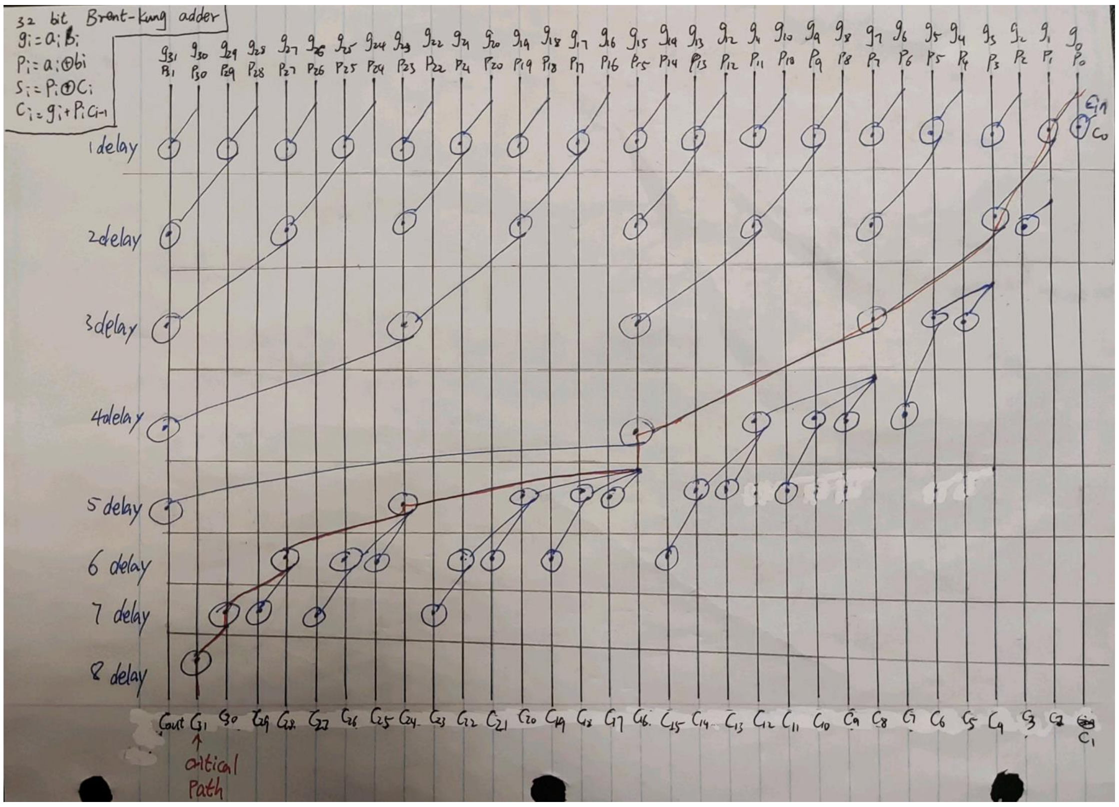
Details

For the structure of the Brent-Kung adder, please refer to the course notes.

Two code files are provided for your as reference. The first one, "**cla_64bit.v**" is a code file for a 64 bit hierarchical carry look ahead adder. You can take this a reference for your code design for you Brent-Kung adder coding. The second one, "**bk_adder_32_tb.v**" is a testbench file for your design. Please read the code carefully.

Questions on RTL/ Behavior Simulation

Q.1. Please plot the structure of the 32-bit Brent_Kung adder and analyse its critical path.



Q.2. Finish the code for the 32-bit Brent_Kung adder (i.e. bk_adder_32bit.v).

```
// ==============================================================================
// Filename: bk_adder_32bit.v
// -----------------------------------------------------------------------------
// This file implement a 32 bits brent kung adder.
// ==============================================================================
`timescale 1 ns / 1 ps
module PG_generator (input wire a,b,
        output wire p,g);
        assign #0.1 p = a ^ b;
        assign #0.1 g = a && b;
endmodule
module dot_operator (input wire gi1,pi1,gi2,pi2,
        output wire go,po);
        assign #0.1 go = gi2 || (pi2 && gi1);
        assign #0.1 po = pi2 && pi1;
endmodule
module bk_adder_32bit #(parameter WIDTH = 32)(
input wire [WIDTH-1:0] a,b,
input wire cin,
output wire [WIDTH-1:0] sum,
output wire cout);

wire [WIDTH-1:0] p_i,g_i;
//level1 means 1 dot operater delay level, and levelN measn N dot operator delay level...
wire [WIDTH-1:0] level1_g, level1_p;
wire [WIDTH-1:2] level2_g, level2_p;
wire [WIDTH-1:4] level3_g, level3_p;
wire [WIDTH-1:6] level4_g, level4_p;
wire [WIDTH-1:10] level5_g, level5_p;
wire [WIDTH-2:14] level6_g, level6_p;
wire [WIDTH-2:22] level7_g, level7_p;
wire level8_g_30, level8_p_30;
wire [WIDTH-2:0] c_i;

//value that will not be synthesize in netlist
genvar i;
//use PG_generator to generate P[31:0] and G[31:0]
generate
    for(i=0;i<32;i=i+1)
        begin: PG_generate
            PG_generator PG(a[i],b[i],p_i[i],g_i[i]);
        end
endgenerate

//level1_dot_operator
    //include the carry_in
assign level1_g[0] = cin;      //level1_g[0]=g+p*g_i[0]=cin+1*(a0 and b0)
assign level1_p[0] = 0;        //level1_p[0]=p*p_i[0]=1*(a0 xor b0)
    //first case dot operater
dot_operator DO(level1_g[0],level1_p[0],g_i[1],p_i[1],level1_g[1],level1_p[1]);
    //forword tree dot operator
generate
    for(i=3;i<32;i=i+2)
        begin: level1_dot_operator_forward_tree
            dot_operator DO1(g_i[i-1],p_i[i-1],g_i[i],p_i[i],level1_g[i],level1_p[i]);
        end
    //assign p_i and g_i to level1_p and level1_g
    for(i=2;i<32;i=i+2)
        begin: pg_connect_level1
            assign level1_p[i] = p_i[i];
            assign level1_g[i] = g_i[i];
        end
endgenerate
assign c_i[0] = cin;
assign c_i[1] = level1_g[0];
assign c_i[2] = level1_g[1];
```

```
//level2_dot_operator
    //reverse tree dot operator
dot_operator RTDO2(level1_g[1],level1_p[1],level1_g[2],level1_p[2],level2_g[2],level2_p[2]);
    //forword tree dot operator
generate
    for(i=3;i<32;i=i+4)
        begin: level2_dot_operator_forward_tree
            dot_operator DO2(level1_g[i-2],level1_p[i-2],level1_g[i],level1_p[i],level2_g[i],level2_p[i]);
        end
    //assign level1_p and level1_g to level2_p and level2_g
    for(i=4;i<7;i=i+1)
        begin: level1_connect_level2
            assign level2_p[i] = level1_p[i];
            assign level2_g[i] = level1_g[i];
        end
    for(i=8;i<11;i=i+1)
        begin: level1_connect1_level2
            assign level2_p[i] = level1_p[i];
            assign level2_g[i] = level1_g[i];
        end

    for(i=12;i<15;i=i+1)
        begin: level1_connect2_level2
            assign level2_p[i] = level1_p[i];
            assign level2_g[i] = level1_g[i];
        end
    for(i=16;i<19;i=i+1)
        begin: level1_connect3_level2
            assign level2_p[i] = level1_p[i];
            assign level2_g[i] = level1_g[i];
        end
    for(i=20;i<23;i=i+1)
        begin: level1_connect4_level2
            assign level2_p[i] = level1_p[i];
            assign level2_g[i] = level1_g[i];
        end
    for(i=24;i<27;i=i+1)
        begin: level1_connect5_level2
            assign level2_p[i] = level1_p[i];
            assign level2_g[i] = level1_g[i];
        end
    for(i=28;i<31;i=i+1)
        begin: level1_connect6_level2
            assign level2_p[i] = level1_p[i];
            assign level2_g[i] = level1_g[i];
```

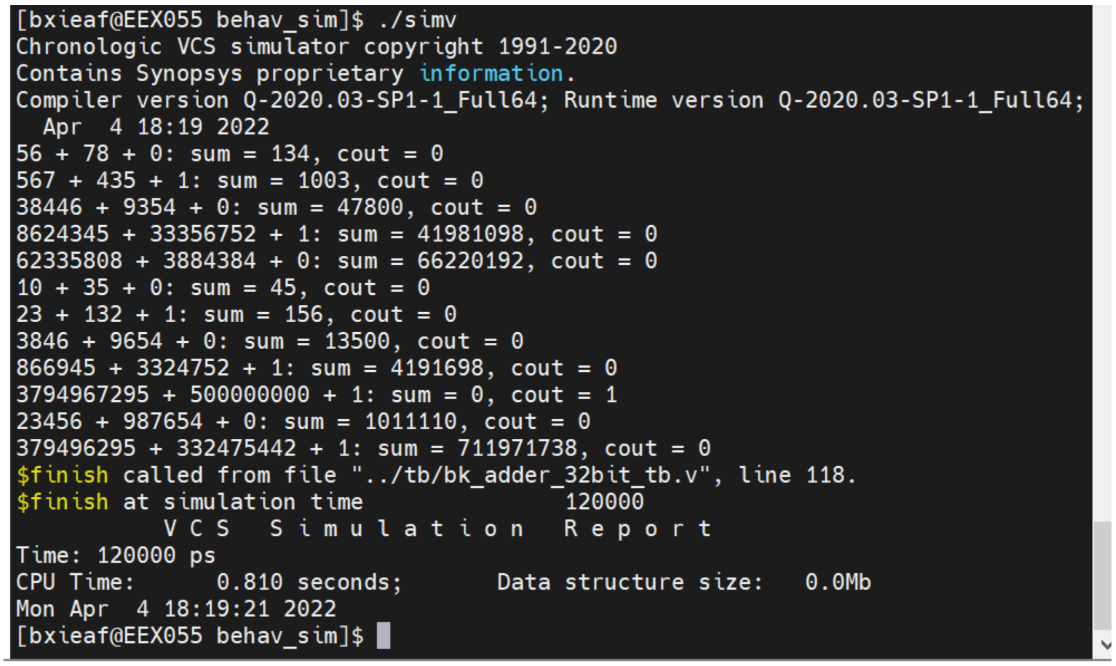
```
                end
endgenerate
//generate carry_i
assign c_i[3] = level2_g[2];
assign c_i[4] = level2_g[3];

//level3_dot_operator
        //reverse tree dot operator
dot_operator RTDO3(level2_g[3],level2_p[3],level2_g[4],level2_p[4],level3_g[4],level3_p[4]);
dot_operator RTDO3_1(level2_g[3],level2_p[3],level2_g[5],level2_p[5],level3_g[5],level3_p[5]);
        //forword tree dot operator
generate
        for(i=7;i<32;i=i+8)
                begin: level3_dot_operator_forward_tree
                        dot_operator DO3(level2_g[i-4],level2_p[i-4],level2_g[i],level2_p[i],level3_g[i],level3_p[i]);
                end
        //assign level2_p and level2_g to level3_p and level3_g
        assign level3_p[6] = level2_p[6];
        assign level3_g[6] = level2_g[6];
        for(i=8;i<15;i=i+1)
                begin: level2_connect_level3
                        assign level3_p[i] = level2_p[i];
                        assign level3_g[i] = level2_g[i];
                end
        for(i=16;i<23;i=i+1)
                begin: level2_connect1_level3
                        assign level3_p[i] = level2_p[i];
                        assign level3_g[i] = level2_g[i];
                end
        for(i=24;i<31;i=i+1)
                begin: level2_connect2_level3
                        assign level3_p[i] = level2_p[i];
                        assign level3_g[i] = level2_g[i];
                end
endgenerate
//generate carry_i
assign c_i[5] = level3_g[4];
assign c_i[6] = level3_g[5];

//level4_dot_operator
        //reverse tree dot operator
dot_operator RTDO4(level3_g[5],level3_p[5],level3_g[6],level3_p[6],level4_g[6],level4_p[6]);
dot_operator RTDO4_1(level3_g[7],level3_p[7],level3_g[8],level3_p[8],level4_g[8],level4_p[8]);
dot_operator RTDO4_2(level3_g[7],level3_p[7],level3_g[9],level3_p[9],level4_g[9],level4_p[9]);
dot_operator RTDO4_3(level3_g[7],level3_p[7],level3_g[11],level3_p[11],level4_g[11],level4_p[11]);
        //forword tree dot operator
```

```
generate
        dot_operator DO4(level3_g[7],level3_p[7],level3_g[15],level3_p[15],level4_g[15],level4_p[15]);
        dot_operator DO4_1(level3_g[23],level3_p[23],level3_g[31],level3_p[31],level4_g[31],level4_p[31]);
        //assign level3_p and level3_g to level4_p and level4_g
        assign level4_p[7] = level3_p[7];
        assign level4_g[7] = level3_g[7];
        assign level4_p[10] = level3_p[10];
        assign level4_g[10] = level3_g[10];
        for(i=12;i<15;i=i+1)
                begin: level3_connect_level4
                        assign level4_p[i] = level3_p[i];
                        assign level4_g[i] = level3_g[i];
                end
        for(i=16;i<31;i=i+1)
                begin: level3_connect1_level4
                        assign level4_p[i] = level3_p[i];
                        assign level4_g[i] = level3_g[i];
                end
endgenerate
//generate carry_i
assign c_i[7] = level4_g[6];
assign c_i[8] = level4_g[7];
assign c_i[9] = level4_g[8];
assign c_i[10] = level4_g[9];

//level5_dot_operator
        //reverse tree dot operator
dot_operator RTDO5(level4_g[9],level4_p[9],level4_g[10],level4_p[10],level5_g[10],level5_p[10]);
dot_operator RTDO5_1(level4_g[11],level4_p[11],level4_g[12],level4_p[12],level5_g[12],level5_p[12]);
dot_operator RTDO5_2(level4_g[11],level4_p[11],level4_g[13],level4_p[13],level5_g[13],level5_p[13]);
dot_operator RTDO5_3(level4_g[15],level4_p[15],level4_g[16],level4_p[16],level5_g[16],level5_p[16]);
dot_operator RTDO5_4(level4_g[15],level4_p[15],level4_g[17],level4_p[17],level5_g[17],level5_p[17]);
dot_operator RTDO5_5(level4_g[15],level4_p[15],level4_g[19],level4_p[19],level5_g[19],level5_p[19]);
dot_operator RTDO5_6(level4_g[15],level4_p[15],level4_g[23],level4_p[23],level5_g[23],level5_p[23]);
        //forword tree dot operator
generate
dot_operator DO5(level4_g[15],level4_p[15],level4_g[31],level4_p[31],level5_g[31],level5_p[31]);
        //assign level4_p and level4_g to level5_p and level5_g
        assign level5_p[11] = level4_p[11];
        assign level5_g[11] = level4_g[11];
        assign level5_p[14] = level4_p[14];
        assign level5_g[14] = level4_g[14];
        assign level5_p[15] = level4_p[15];
        assign level5_g[15] = level4_g[15];
        assign level5_p[18] = level4_p[18];
        assign level5_g[18] = level4_g[18];
        for(i=20;i<23;i=i+1)
                begin: level4_connect_level5
                        assign level5_p[i] = level4_p[i];
                        assign level5_g[i] = level4_g[i];
                end
        for(i=24;i<31;i=i+1)
                begin: level4_connect1_level5
                        assign level5_p[i] = level4_p[i];
                        assign level5_g[i] = level4_g[i];
                end
endgenerate
//generate carry_i
assign c_i[11] = level5_g[10];
assign c_i[12] = level5_g[11];
assign c_i[13] = level5_g[12];
assign c_i[14] = level5_g[13];
//generate carry_out
assign cout = level5_g[31];

//level6_dot_operator
        //reverse tree dot operator
dot_operator RTDO6(level5_g[13],level5_p[13],level5_g[14],level5_p[14],level6_g[14],level6_p[14]);
dot_operator RTDO6_1(level5_g[17],level5_p[17],level5_g[18],level5_p[18],level6_g[18],level6_p[18]);
dot_operator RTDO6_2(level5_g[19],level5_p[19],level5_g[20],level5_p[20],level6_g[20],level6_p[20]);
dot_operator RTDO6_3(level5_g[19],level5_p[19],level5_g[21],level5_p[21],level6_g[21],level6_p[21]);
dot_operator RTDO6_4(level5_g[23],level5_p[23],level5_g[24],level5_p[24],level6_g[24],level6_p[24]);
dot_operator RTDO6_5(level5_g[23],level5_p[23],level5_g[25],level5_p[25],level6_g[25],level6_p[25]);
dot_operator RTDO6_6(level5_g[23],level5_p[23],level5_g[27],level5_p[27],level6_g[27],level6_p[27]);
        //assign level5_p and level5_g to level6_p and level6_g
generate
        for(i=15;i<18;i=i+1)
                begin: level5_connect_level6
                        assign level6_p[i] = level5_p[i];
                        assign level6_g[i] = level5_g[i];
                end
        assign level6_p[19] = level5_p[19];
        assign level6_g[19] = level5_g[19];
        assign level6_p[22] = level5_p[22];
        assign level6_g[22] = level5_g[22];
        assign level6_p[23] = level5_p[23];
        assign level6_g[23] = level5_g[23];
        assign level6_p[26] = level5_p[26];
        assign level6_g[26] = level5_g[26];
        for(i=28;i<31;i=i+1)
                begin: level5_connect1_level6
                        assign level6_p[i] = level5_p[i];
                        assign level6_g[i] = level5_g[i];
                end
endgenerate
//generate carry_i
assign c_i[15] = level6_g[14];
assign c_i[16] = level6_g[15];
assign c_i[17] = level6_g[16];
assign c_i[18] = level6_g[17];
assign c_i[19] = level6_g[18];
assign c_i[20] = level6_g[19];
assign c_i[21] = level6_g[20];
assign c_i[22] = level6_g[21];

//level7_dot_operator
        //reverse tree dot operator
dot_operator RTDO7(level6_g[21],level6_p[21],level6_g[22],level6_p[22],level7_g[22],level7_p[22]);
dot_operator RTDO7_1(level6_g[25],level6_p[25],level6_g[26],level6_p[26],level7_g[26],level7_p[26]);
dot_operator RTDO7_2(level6_g[27],level6_p[27],level6_g[28],level6_p[28],level7_g[28],level7_p[28]);
dot_operator RTDO7_3(level6_g[27],level6_p[27],level6_g[29],level6_p[29],level7_g[29],level7_p[29]);
        //assign level6_p and level6_g to level7_p and level7_g
generate
        for(i=23;i<26;i=i+1)
                begin: level6_connect_level7
                        assign level7_p[i] = level6_p[i];
                        assign level7_g[i] = level6_g[i];
                end
        assign level7_p[27] = level6_p[27];
        assign level7_g[27] = level6_g[27];
        assign level7_p[30] = level6_p[30];
        assign level7_g[30] = level6_g[30];
endgenerate
//generate carry_i
assign c_i[23] = level7_g[22];
assign c_i[24] = level7_g[23];
assign c_i[25] = level7_g[24];
assign c_i[26] = level7_g[25];
assign c_i[27] = level7_g[26];
assign c_i[28] = level7_g[27];
assign c_i[29] = level7_g[28];
assign c_i[30] = level7_g[29];

//level8_dot_operator
        //reverse tree dot operator
dot_operator RTDO8(level7_g[29],level7_p[29],level7_g[30],level7_p[30],level8_g_30,level8_p_30);
//generate carry_i
assign c_i[31] = level8_g_30;
```

```
//generate sum
generate
        for(i=0;i<31;i=i+1)
        begin: sum_generator
                xor #0.1 (sum[i], p_i[i], c_i[i]);
        end
        xor #0.1 (sum[31], p_i[31], cout);
endgenerate
endmodule
```

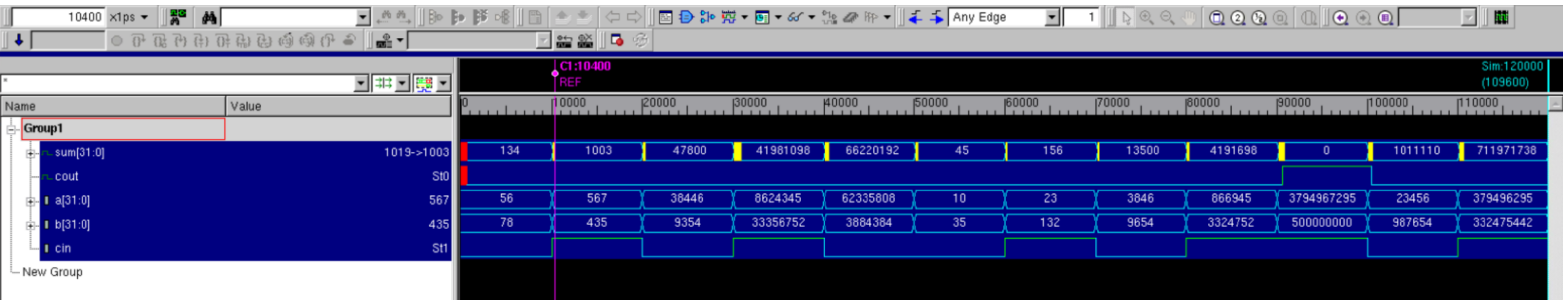
Q.3. Run the behavior-level simulation of your design and check the result printed in the terminal. A sample result is shown as below.

Then, modify the input stimulus in bk_adder_32_tb.v. (You should hand in your own screenshot of behavior simulation. **<u>Different</u>** input stimulus must be applied to your design to generate a different result.) Report the result you generated. When you take the screenshot, please include your username in it.



```
[bxieaf@EEX055 behav_sim]$ ./simv
Chronologic VCS simulator copyright 1991-2020
Contains Synopsys proprietary information.
Compiler version Q-2020.03-SP1-1_Full64; Runtime version Q-2020.03-SP1-1_Full64;
  Apr  4 18:19 2022
56 + 78 + 0: sum = 134, cout = 0
567 + 435 + 1: sum = 1003, cout = 0
38446 + 9354 + 0: sum = 47800, cout = 0
8624345 + 33356752 + 1: sum = 41981098, cout = 0
62335808 + 3884384 + 0: sum = 66220192, cout = 0
10 + 35 + 0: sum = 45, cout = 0
23 + 132 + 1: sum = 156, cout = 0
3846 + 9654 + 0: sum = 13500, cout = 0
866945 + 3324752 + 1: sum = 4191698, cout = 0
3794967295 + 500000000 + 1: sum = 0, cout = 1
23456 + 987654 + 0: sum = 1011110, cout = 0
379496295 + 332475442 + 1: sum = 711971738, cout = 0
$finish called from file "../tb/bk_adder_32bit_tb.v", line 118.
$finish at simulation time               120000
           V C S   S i m u l a t i o n   R e p o r t
Time: 120000 ps
CPU Time:      0.810 seconds;      Data structure size:   0.0Mb
Mon Apr  4 18:19:21 2022
[bxieaf@EEX055 behav_sim]$
```

Q.4. Please report the waveform based on your input stimulus. To check the waveform, use the command "./simv -gui". After opening DVE, please select the signals, right click, and select "Add Waveforms". Then, press the ↓ button marked in the following figure to run the simulation, and you will get the waveform. Before taking the screenshot, please change the radix from hexadecimal to decimal by selecting the signals, right click, and select "Set Radix".



Questions on Synthesis

We will use the Nangate FreePDK 45nm as the sampled PDK for our design. For your convenience, the library has already been included in the UST ECE servers. Please check the /afs/ee.ust.hk/staff/ee/dept/public/elec516/eesm_5020_2017spring/eesm_5020/lib/.

You don't need to download the PDK or do the library conversion (from *.lib to *.db), because we have already done that for you. In this assignment, you can use the library "**NangateOpenCellLibrary_slow.db**" for synthesis.

Q.5. We need to understand the library information before running our synthesis. Complete the following table about Nangate FreePDK 45nm. [Hint: you can find the information in "lib/NangateOpenCellLibrary_PDKv1_3_v2010_12/Front_End/Liberty/NLDM" in the sample project on Canvas. Use the command "less NangateOpenCellLibrary_slow.lib" to read the details. Then, type "q" to exit.]

Nangate FreePDK 45nm library information	
Area unit	1um2
Time unit	1ns
Leakage power unit	1nW
Voltage unit	1V
Capacitance unit	1fF

The area of the 2-input NAND gate of unit drive strength is an important metric to derive the equivalent gate number of the circuit. Please find out the area of the 2-input

NAND gate. [Hint: you can find the information in the files under the 'lib' folder on UST ECE servers: "/lib/NangateOpenCellLibrary_PDKv1_3_v2010_12/Front_End/Liberty". Use the command "less NangateOpenCellLibrary_functional.lib" to read the details. Then, type "q" to exit.]

Nangate FreePDK 45nm 2-input NAND gate (Cell name: NAND2_X1)	
Area [um2]	0.798 um2

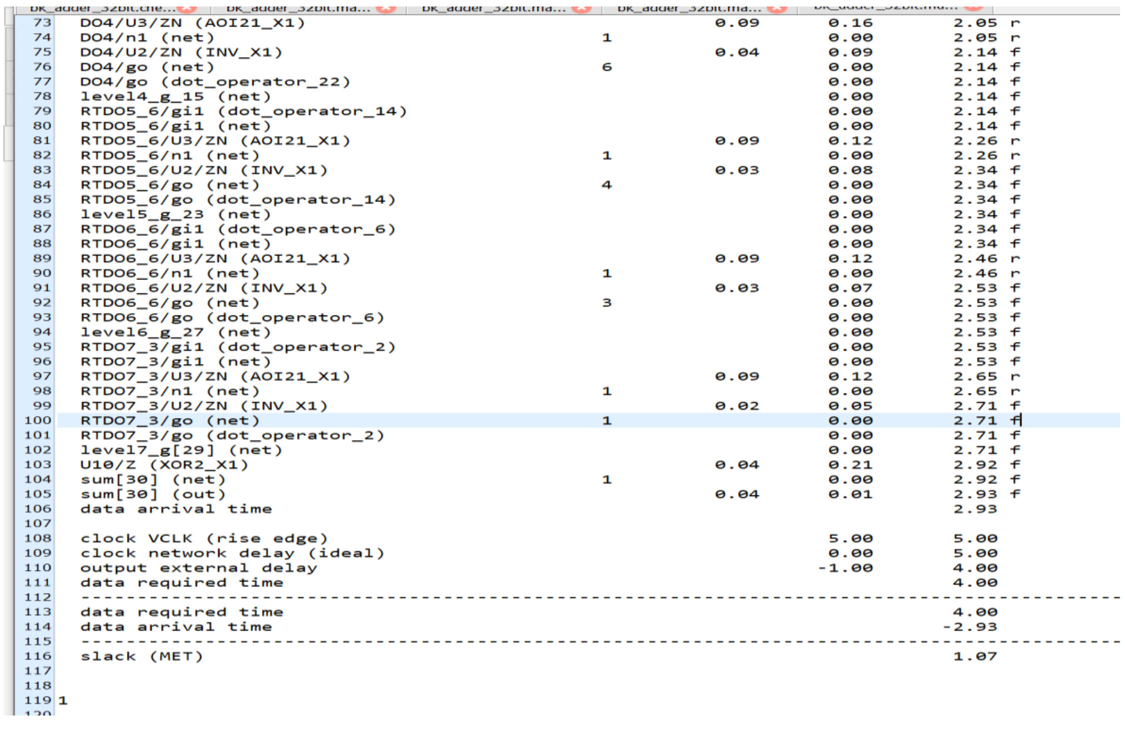
Q.6. Modify the provided scripts to run the whole synthesis flow. [Hint: syn/run.tcl contains the library setup. The library location may be different from your system. Please modify the script accordingly. Note that since the designed module is a pure combinational logic, we need to create a virtual clock to constraint the time. Please refer to page 82 of the Verilog tutorial slides for how to create a virtual clock.] After successfully running the synthesis, you should be able to get different reports for your design.

Please complete the following summary table of your design:

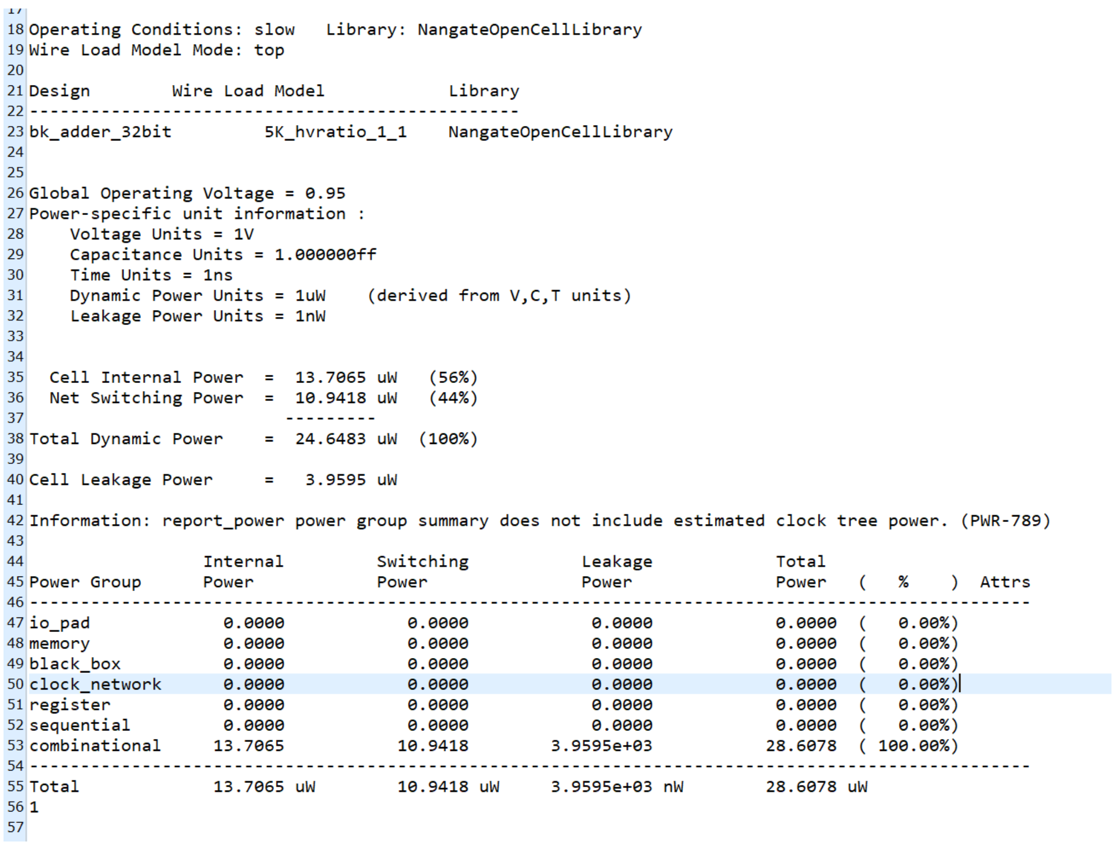
Brent_Kung adder synthesized under timing constraint = 5ns	
Area	252.168 um2
Critical path [Hint 1]	a[15] to sum[30] =2.93 ns slack (1.07ns)
Leakage power	3.9595 uW
Dynamic power [Hint 2]	24.6483 uW
Total power	28.6078 uW

Notice: Input delay(max,min)=(1ns,1ns), output delay(max, min)=(1ns,1ns)

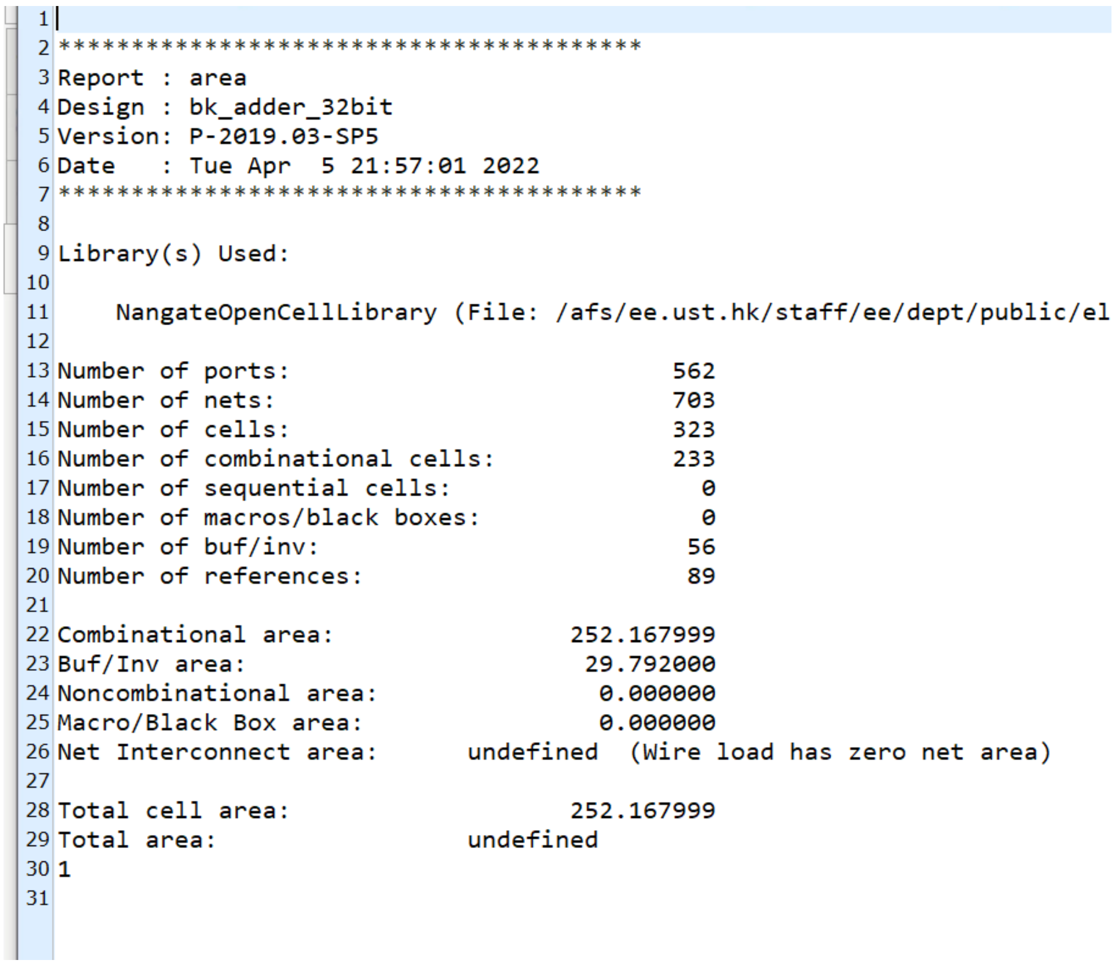
Here is the report for above table:



```
DO4/U3/ZN (AOI21_X1)                                 0.09      0.16       2.05 r
DO4/n1 (net)                                1                  0.00       2.05 r
DO4/U2/ZN (INV_X1)                                   0.04      0.09       2.14 f
DO4/go (net)                                6                  0.00       2.14 f
DO4/go (dot_operator_22)                                       0.00       2.14 f
level4_g_15 (net)                                              0.00       2.14 f
RTDO5_6/gi1 (dot_operator_14)                                  0.00       2.14 f
RTDO5_6/gi1 (net)                                              0.00       2.14 f
RTDO5_6/U3/ZN (AOI21_X1)                             0.09      0.12       2.26 r
RTDO5_6/n1 (net)                            1                  0.00       2.26 r
RTDO5_6/U2/ZN (INV_X1)                               0.03      0.08       2.34 f
RTDO5_6/go (net)                            4                  0.00       2.34 f
RTDO5_6/go (dot_operator_14)                                   0.00       2.34 f
level5_g_23 (net)                                              0.00       2.34 f
RTDO6_6/gi1 (dot_operator_6)                                   0.00       2.34 f
RTDO6_6/gi1 (net)                                              0.00       2.34 f
RTDO6_6/U3/ZN (AOI21_X1)                             0.09      0.12       2.46 r
RTDO6_6/n1 (net)                            1                  0.00       2.46 r
RTDO6_6/U2/ZN (INV_X1)                               0.03      0.07       2.53 f
RTDO6_6/go (net)                            3                  0.00       2.53 f
RTDO6_6/go (dot_operator_6)                                    0.00       2.53 f
level6_g_27 (net)                                              0.00       2.53 f
RTDO7_3/gi1 (dot_operator_2)                                   0.00       2.53 f
RTDO7_3/gi1 (net)                                              0.00       2.53 f
RTDO7_3/U3/ZN (AOI21_X1)                             0.09      0.12       2.65 r
RTDO7_3/n1 (net)                            1                  0.00       2.65 r
RTDO7_3/U2/ZN (INV_X1)                               0.02      0.05       2.71 f
RTDO7_3/go (net)                            1                  0.00       2.71 f
RTDO7_3/go (dot_operator_2)                                    0.00       2.71 f
level7_g[29] (net)                                             0.00       2.71 f
U10/Z (XOR2_X1)                                      0.04      0.21       2.92 f
sum[30] (net)                               1                  0.00       2.92 f
sum[30] (out)                                        0.04      0.01       2.93 f
data arrival time                                                         2.93

clock VCLK (rise edge)                                         5.00       5.00
clock network delay (ideal)                                    0.00       5.00
output external delay                                         -1.00       4.00
data required time                                                        4.00
---------------------------------------------------------------------------------
data required time                                                        4.00
data arrival time                                                        -2.93
---------------------------------------------------------------------------------
slack (MET)                                                               1.07


1
```



```
Operating Conditions: slow   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
bk_adder_32bit         5K_hvratio_1_1     NangateOpenCellLibrary


Global Operating Voltage = 0.95
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  13.7065 uW   (56%)
  Net Switching Power  =  10.9418 uW   (44%)
                         ---------
Total Dynamic Power    =  24.6483 uW  (100%)

Cell Leakage Power     =   3.9595 uW

Information: report_power power group summary does not include estimated clock tree power. (PWR-789)

                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000            0.0000  (   0.00%)
register           0.0000           0.0000            0.0000            0.0000  (   0.00%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational     13.7065          10.9418        3.9595e+03           28.6078  ( 100.00%)
--------------------------------------------------------------------------------------------------
Total             13.7065 uW       10.9418 uW     3.9595e+03 nW        28.6078 uW
1
```



```

****************************************
Report : area
Design : bk_adder_32bit
Version: P-2019.03-SP5
Date   : Tue Apr  5 21:57:01 2022
****************************************

Library(s) Used:

    NangateOpenCellLibrary (File: /afs/ee.ust.hk/staff/ee/dept/public/el

Number of ports:                          562
Number of nets:                           703
Number of cells:                          323
Number of combinational cells:            233
Number of sequential cells:                 0
Number of macros/black boxes:               0
Number of buf/inv:                         56
Number of references:                      89

Combinational area:                252.167999
Buf/Inv area:                       29.792000
Noncombinational area:               0.000000
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                   252.167999
Total area:                 undefined
1
```

Q.7. Modify the timing constraint to different values and re-run the synthesis under different constraints. [Hint: the timing constraint is defined in divider.constraints.tcl in the sample project on Canvas. You need to modify the file.]

Brent_Kung adder synthesized in Nangate 45nm			
Timing constraint	2ns	5ns	10ns
Area	Can we meet the time constraint? [Y/[✓ N]]	252.168 um2	252.168 um2
Critical path		2.93 ns (slack 1.07ns)	2.93 ns (slack 6.07ns)
Leakage power		3.9595 uW	3.9595 uW
Dynamic power		24.65 uW	12.3 uW
Total power		28.6 uW	16.28 uW

Notice: Input delay(max,min)=(1ns,1ns), output delay(max, min)=(1ns,1ns)

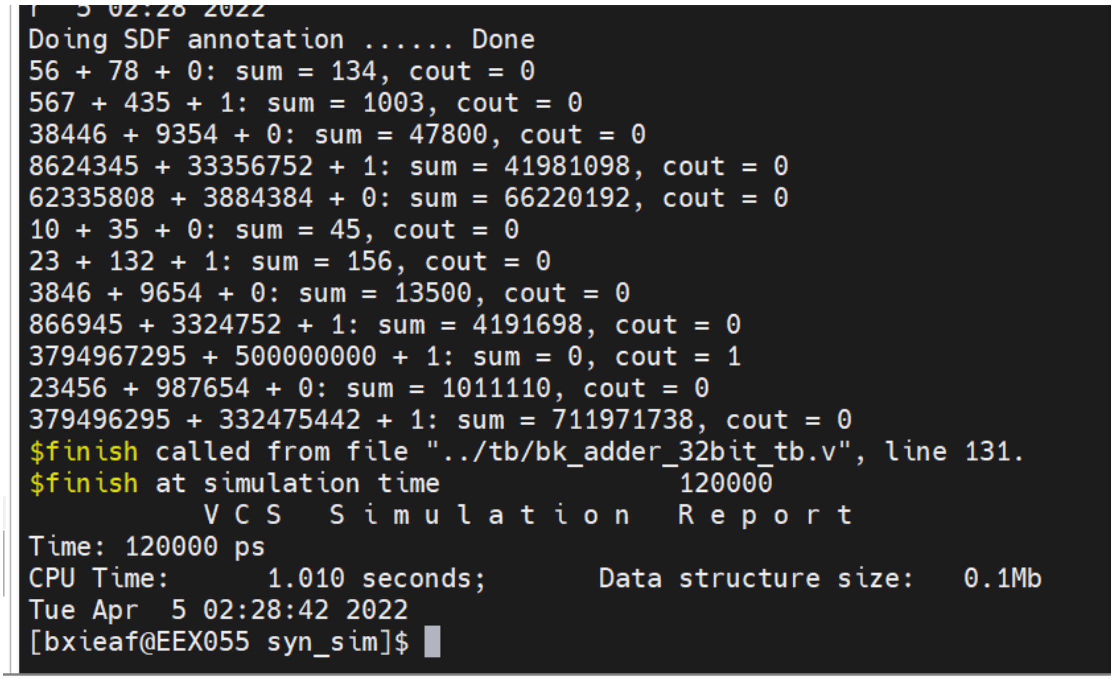
Analysis: Small Timing constraint mean high frequency. When 1 unit of time period equals to 2ns, it is easy to have time violation. Thus, 2ns is not suitable in this case. 5ns and 10ns are suitable for the design, however, higher frequency consumes more power. So, 5ns require more power to compute. The critical path delay is 2.93ns in both suitable case but the critical path slack is obviously different. Last, the leakage power is identical for both cases.

(Information to be included for Q.7: one table and brief analysis.)

Questions on Post-synthesis simulation

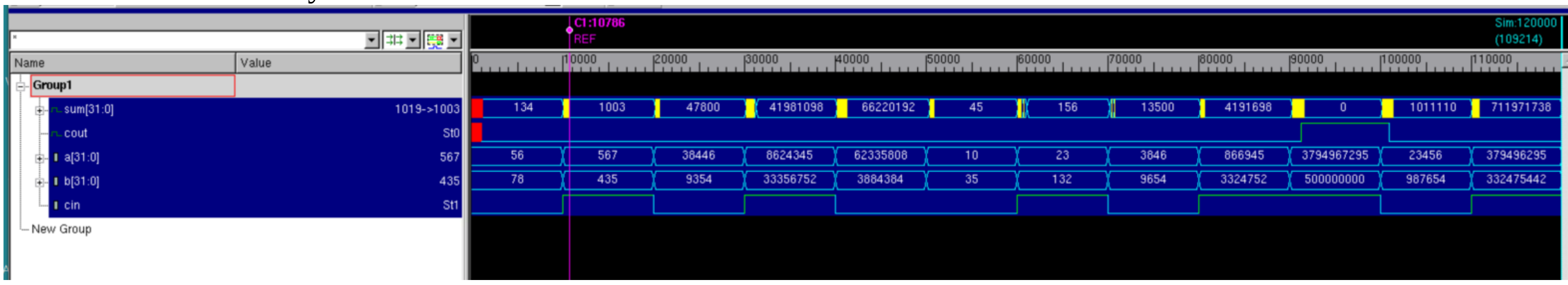
For the following questions, you only need to run the simulation and P&R on the 32-bit Brent_Kung adder synthesized under 5ns timing constraint.

Q.8. Run the post synthesis simulation. You need to modify the VCS compilation scripts (i.e. run) according to your settings. Please also remember to modify your testbench to annotate the information in the "sdf" file.

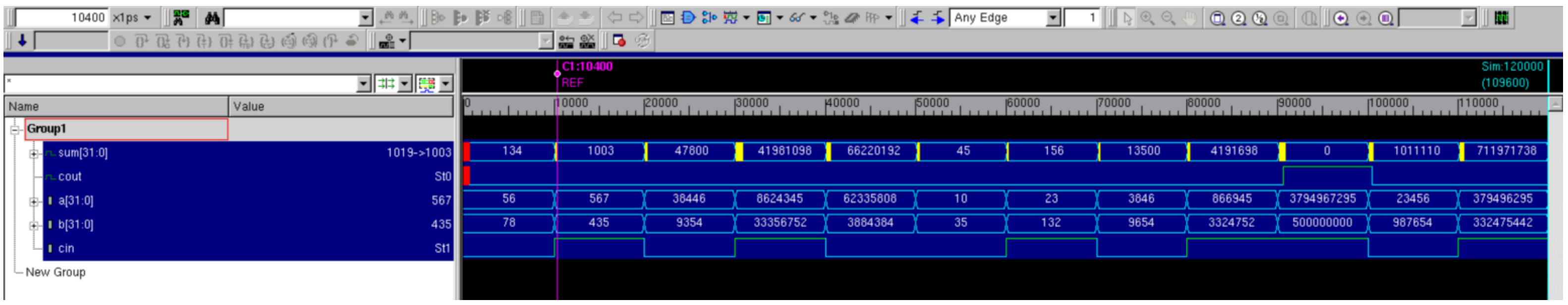


```
r  5 02:28 2022
Doing SDF annotation ...... Done
56 + 78 + 0: sum = 134, cout = 0
567 + 435 + 1: sum = 1003, cout = 0
38446 + 9354 + 0: sum = 47800, cout = 0
8624345 + 33356752 + 1: sum = 41981098, cout = 0
62335808 + 3884384 + 0: sum = 66220192, cout = 0
10 + 35 + 0: sum = 45, cout = 0
23 + 132 + 1: sum = 156, cout = 0
3846 + 9654 + 0: sum = 13500, cout = 0
866945 + 3324752 + 1: sum = 4191698, cout = 0
3794967295 + 500000000 + 1: sum = 0, cout = 1
23456 + 987654 + 0: sum = 1011110, cout = 0
379496295 + 332475442 + 1: sum = 711971738, cout = 0
$finish called from file "../tb/bk_adder_32bit_tb.v", line 131.
$finish at simulation time               120000
           V C S   S i m u l a t i o n   R e p o r t
Time: 120000 ps
CPU Time:      1.010 seconds;       Data structure size:   0.1Mb
Tue Apr  5 02:28:42 2022
[bxieaf@EEX055 syn_sim]$
```

Post-Synthesis Simulation



behavior simulation



Questions on P&R

Do the P&R for the synthesized netlist from Design Compiler.

Q.9. Three reports should be handed after the layout step:

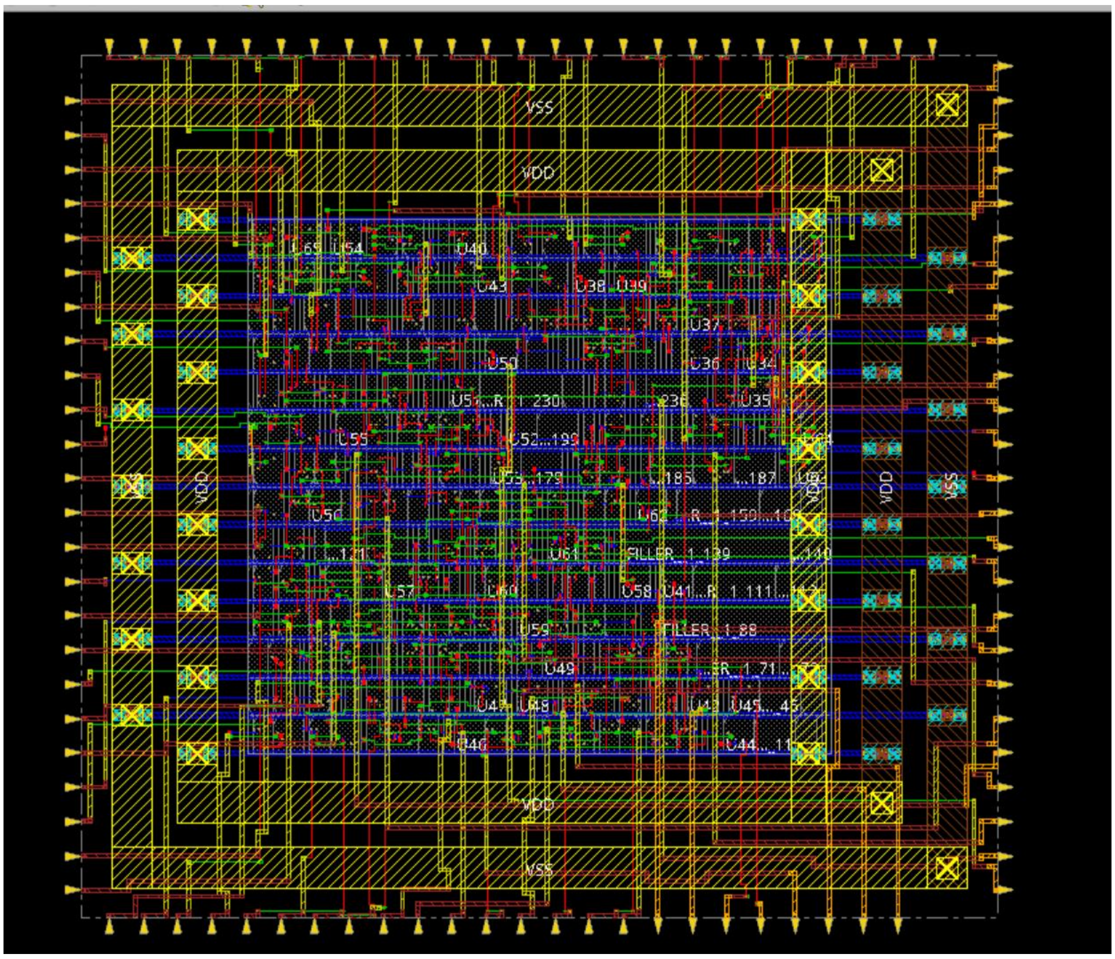
- Final layout view of your Brent_Kung adder

- Setup timing report (histogram) of Brent_Kung adder

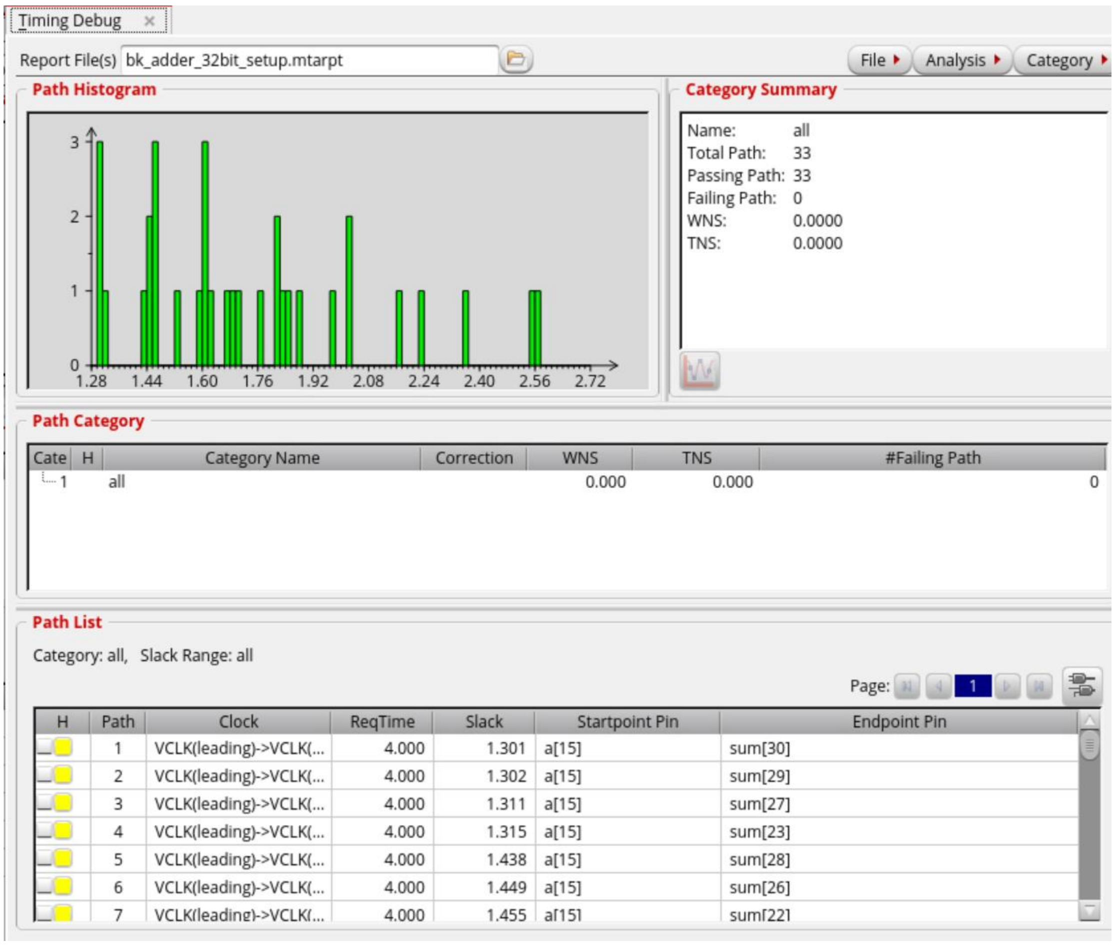
- Hold timing report (histogram) of Brent_Kung adder

Notice: Input delay(max,min)=(1ns,1ns), output delay(max, min)=(1ns,1ns)

<u>Final layout</u>



Setup timing report (histogram) of Brent_Kung adder:



Report File(s) bk_adder_32bit_setup.mtarpt

Path Histogram

Category Summary

Name:	all
Total Path:	33
Passing Path:	33
Failing Path:	0
WNS:	0.0000
TNS:	0.0000

Path Category

Cate	H	Category Name	Correction	WNS	TNS	#Failing Path
1		all		0.000	0.000	0

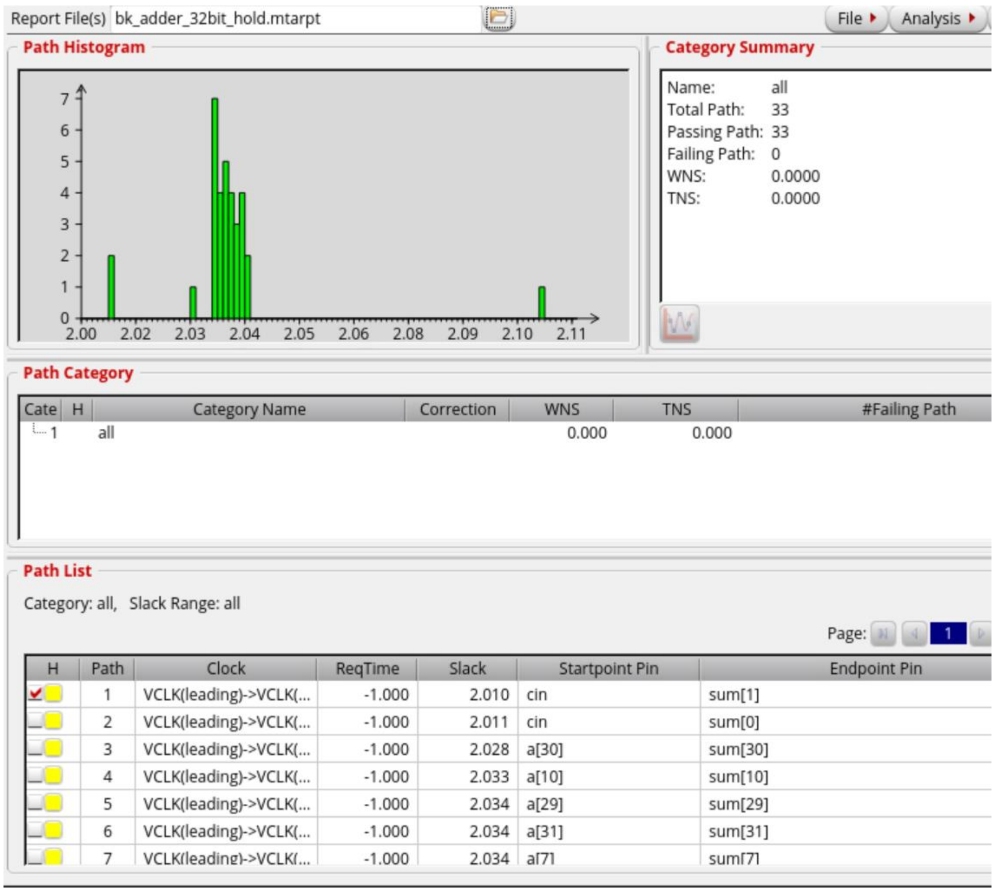
Path List

Category: all, Slack Range: all

Page: 1

H	Path	Clock	ReqTime	Slack	Startpoint Pin	Endpoint Pin
	1	VCLK(leading)->VCLK(...	4.000	1.301	a[15]	sum[30]
	2	VCLK(leading)->VCLK(...	4.000	1.302	a[15]	sum[29]
	3	VCLK(leading)->VCLK(...	4.000	1.311	a[15]	sum[27]
	4	VCLK(leading)->VCLK(...	4.000	1.315	a[15]	sum[23]
	5	VCLK(leading)->VCLK(...	4.000	1.438	a[15]	sum[28]
	6	VCLK(leading)->VCLK(...	4.000	1.449	a[15]	sum[26]
	7	VCLK(leading)->VCLK(...	4.000	1.455	a[15]	sum[22]

Hold timing report (histogram) of Brent_Kung adder



Report File(s) bk_adder_32bit_hold.mtarpt

Path Histogram

Category Summary

Name:	all
Total Path:	33
Passing Path:	33
Failing Path:	0
WNS:	0.0000
TNS:	0.0000

Path Category

Cate	H	Category Name	Correction	WNS	TNS	#Failing Path
1		all		0.000	0.000	

Path List

Category: all, Slack Range: all

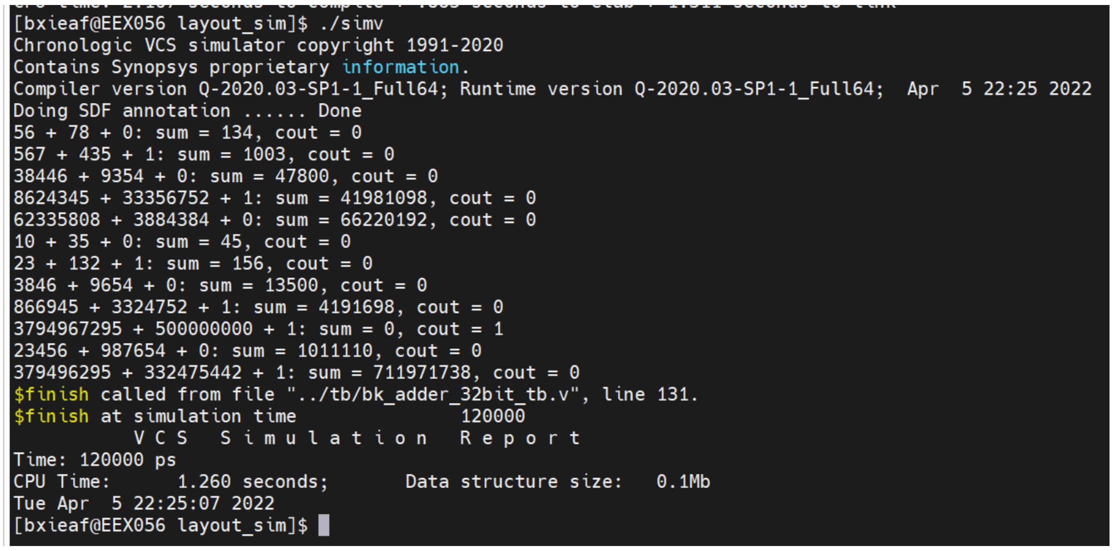
Page: 1

H	Path	Clock	ReqTime	Slack	Startpoint Pin	Endpoint Pin
✓	1	VCLK(leading)->VCLK(...	-1.000	2.010	cin	sum[1]
	2	VCLK(leading)->VCLK(...	-1.000	2.011	cin	sum[0]
	3	VCLK(leading)->VCLK(...	-1.000	2.028	a[30]	sum[30]
	4	VCLK(leading)->VCLK(...	-1.000	2.033	a[10]	sum[10]
	5	VCLK(leading)->VCLK(...	-1.000	2.034	a[29]	sum[29]
	6	VCLK(leading)->VCLK(...	-1.000	2.034	a[31]	sum[31]
	7	VCLK(leading)->VCLK(...	-1.000	2.034	a[7]	sum[7]

Questions on Post-layout simulation

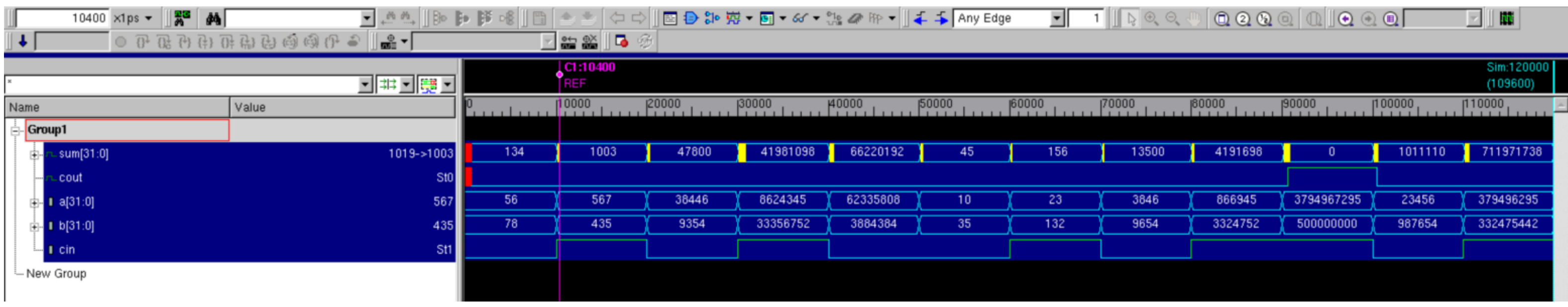
Q.10. Run the post layout simulation under layout_sim directory. As in Q.9, you should include your own waveform of the post-layout simulation.

(Information to be included for Q.10: your own results of the post-layout simulation.)

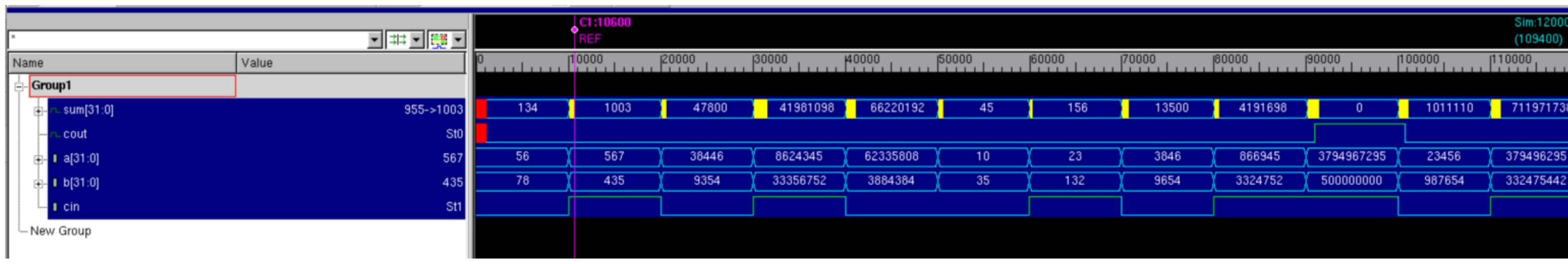


```
[bxieaf@EEX056 layout_sim]$ ./simv
Chronologic VCS simulator copyright 1991-2020
Contains Synopsys proprietary information.
Compiler version Q-2020.03-SP1-1_Full64; Runtime version Q-2020.03-SP1-1_Full64;  Apr  5 22:25 2022
Doing SDF annotation ...... Done
56 + 78 + 0: sum = 134, cout = 0
567 + 435 + 1: sum = 1003, cout = 0
38446 + 9354 + 0: sum = 47800, cout = 0
8624345 + 33356752 + 1: sum = 41981098, cout = 0
62335808 + 3884384 + 0: sum = 66220192, cout = 0
10 + 35 + 0: sum = 45, cout = 0
23 + 132 + 1: sum = 156, cout = 0
3846 + 9654 + 0: sum = 13500, cout = 0
866945 + 3324752 + 1: sum = 4191698, cout = 0
3794967295 + 500000000 + 1: sum = 0, cout = 1
23456 + 987654 + 0: sum = 1011110, cout = 0
379496295 + 332475442 + 1: sum = 711971738, cout = 0
$finish called from file "../tb/bk_adder_32bit_tb.v", line 131.
$finish at simulation time               120000
           V C S   S i m u l a t i o n   R e p o r t
Time: 120000 ps
CPU Time:      1.260 seconds;       Data structure size:   0.1Mb
Tue Apr  5 22:25:07 2022
[bxieaf@EEX056 layout_sim]$
```

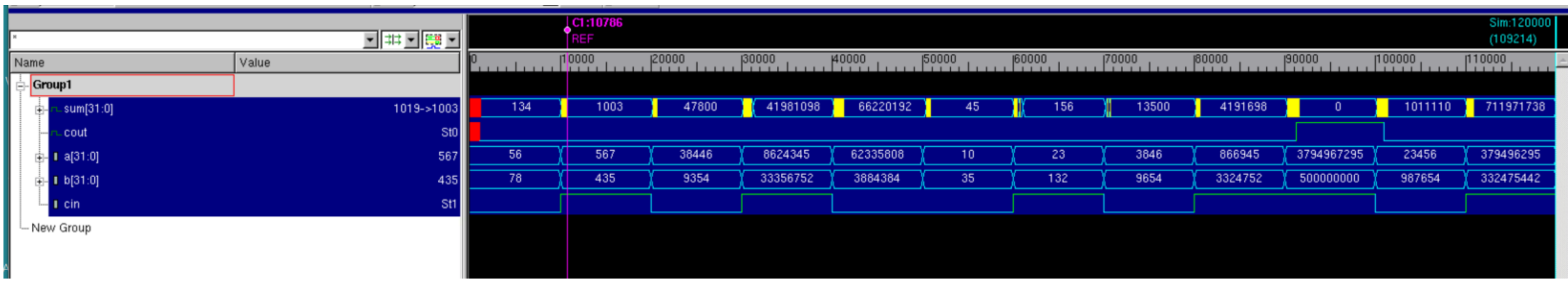
behavior simulation:



Post-Synthesis Simulation :



Post layout simulation:



Questions on Different Types of Adders

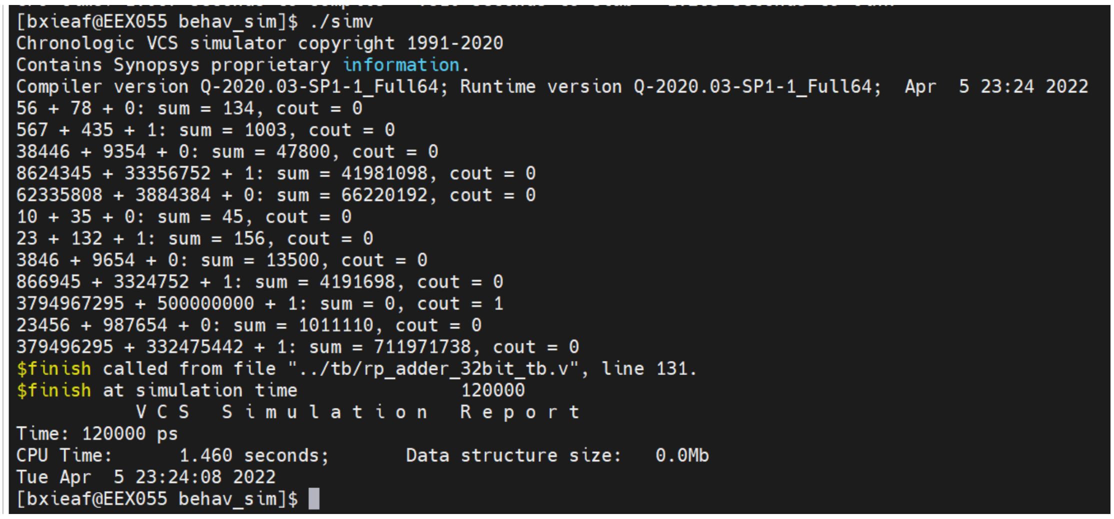
Q.11 Please design a 32-bit ripple carry adder and go through the design flow. Please compare the speed and area of the two 32-bit adder design.

(Information to be included for Q.12: RTL code, waveform of behavior-level simulation, speed and area analysis after synthesis, waveform of post-synthesis simulation, layout, waveform of post-layout simulation)

<u>Verilog code:</u>

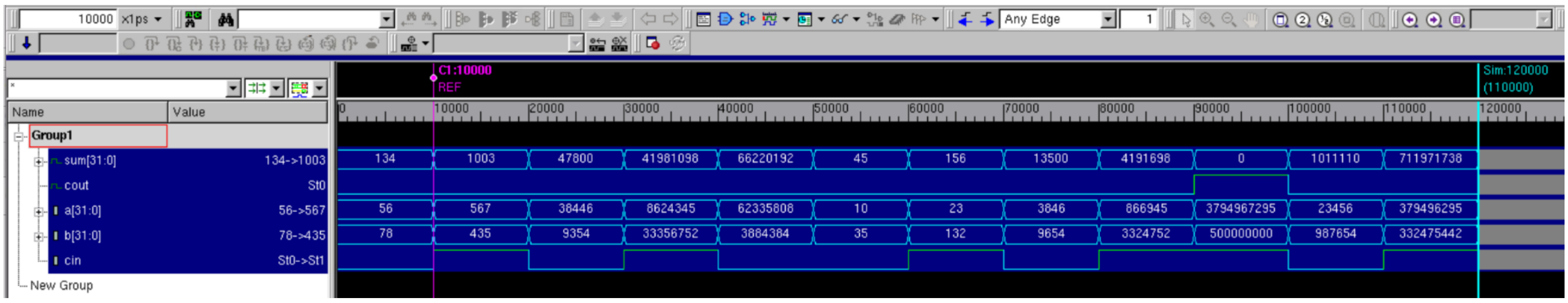
```
//=====================================================================
// Filename: rp_adder_32bit.v
// This file implement a 32 bits ripple carry adder.
//=====================================================================
`timescale 1 ns / 1 ps
module rp_adder_32bit #(parameter WIDTH = 32)(
input wire [WIDTH-1:0] a,b,
input wire cin,
output wire [WIDTH-1:0] sum,
output wire cout);

wire [WIDTH-2:0] c_i;

//first case
assign c_i[0] = a[0]&&b[0]||(cin&&(a[0]||b[0]));
assign sum[0] = a[0] ^ b[0] ^ cin;
genvar i;
generate
	for(i=1;i<31;i=i+1)
		begin: c_s_generator
			assign c_i[i] = a[i]&&b[i]||(c_i[i-1]&&(a[i]||b[i]));
			assign sum[i] = a[i] ^ b[i] ^ c_i[i-1];
		end
endgenerate

//last case
assign cout = a[31]&&b[31]||(c_i[30]&&(a[31]||b[31]));
assign sum[31] = a[31] ^ b[31] ^ c_i[30];
endmodule
```

waveform of behavior-level simulation of ripple carry adder



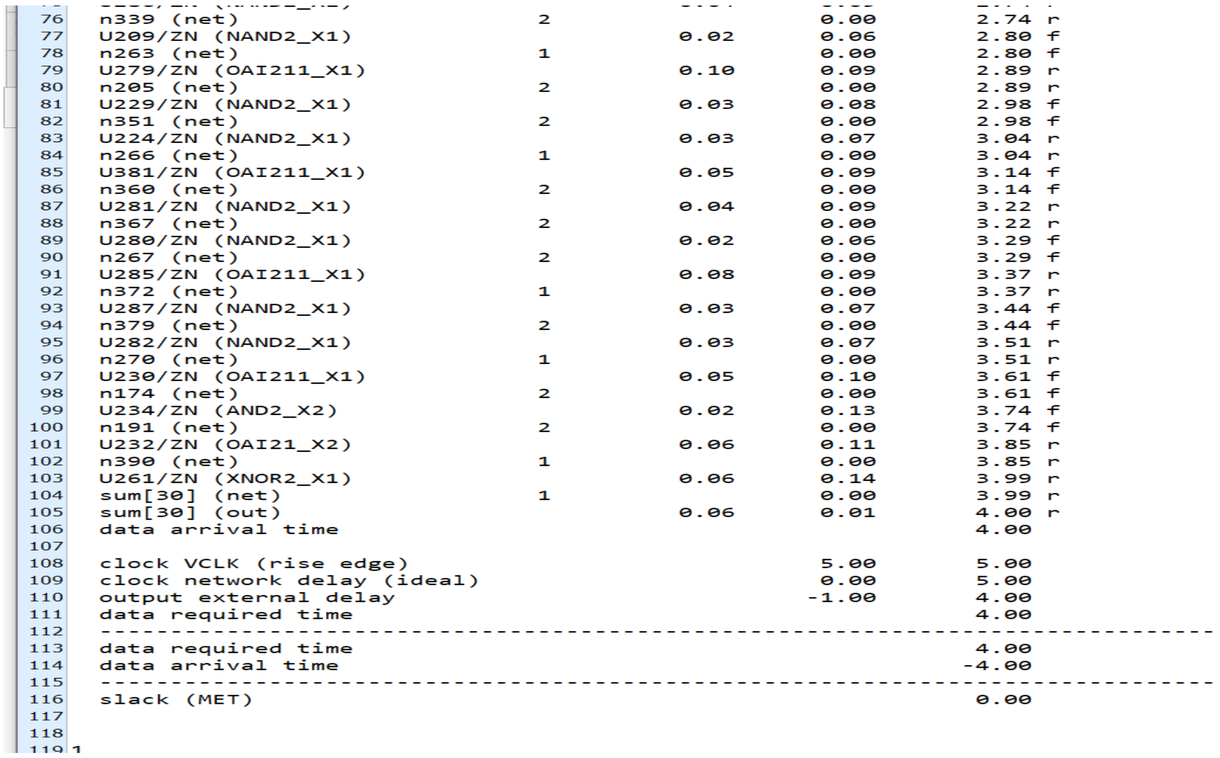
```
[bxieaf@EEX055 behav_sim]$ ./simv
Chronologic VCS simulator copyright 1991-2020
Contains Synopsys proprietary information.
Compiler version Q-2020.03-SP1-1_Full64; Runtime version Q-2020.03-SP1-1_Full64;  Apr  5 23:24 2022
56 + 78 + 0: sum = 134, cout = 0
567 + 435 + 1: sum = 1003, cout = 0
38446 + 9354 + 0: sum = 47800, cout = 0
8624345 + 33356752 + 1: sum = 41981098, cout = 0
62335808 + 3884384 + 0: sum = 66220192, cout = 0
10 + 35 + 0: sum = 45, cout = 0
23 + 132 + 1: sum = 156, cout = 0
3846 + 9654 + 0: sum = 13500, cout = 0
866945 + 3324752 + 1: sum = 4191698, cout = 0
3794967295 + 500000000 + 1: sum = 0, cout = 1
23456 + 987654 + 0: sum = 1011110, cout = 0
379496295 + 332475442 + 1: sum = 711971738, cout = 0
$finish called from file "../tb/rp_adder_32bit_tb.v", line 131.
$finish at simulation time              120000
           V C S   S i m u l a t i o n   R e p o r t
Time: 120000 ps
CPU Time:      1.460 seconds;       Data structure size:   0.0Mb
Tue Apr  5 23:24:08 2022
[bxieaf@EEX055 behav_sim]$
```

behavior simulation:

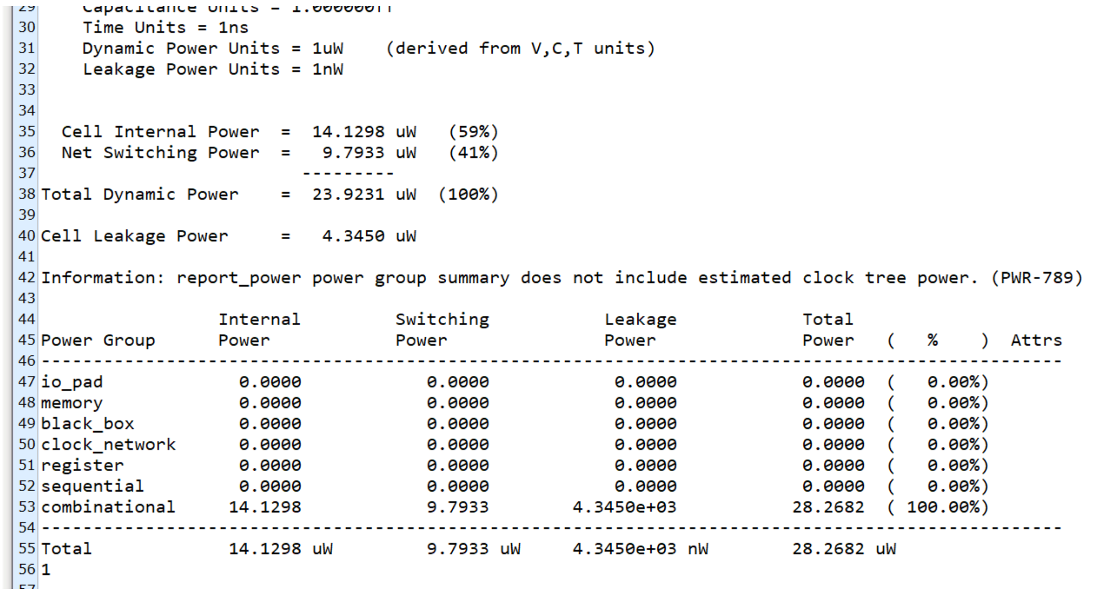


Synthesis report for ripple carry adder

Time

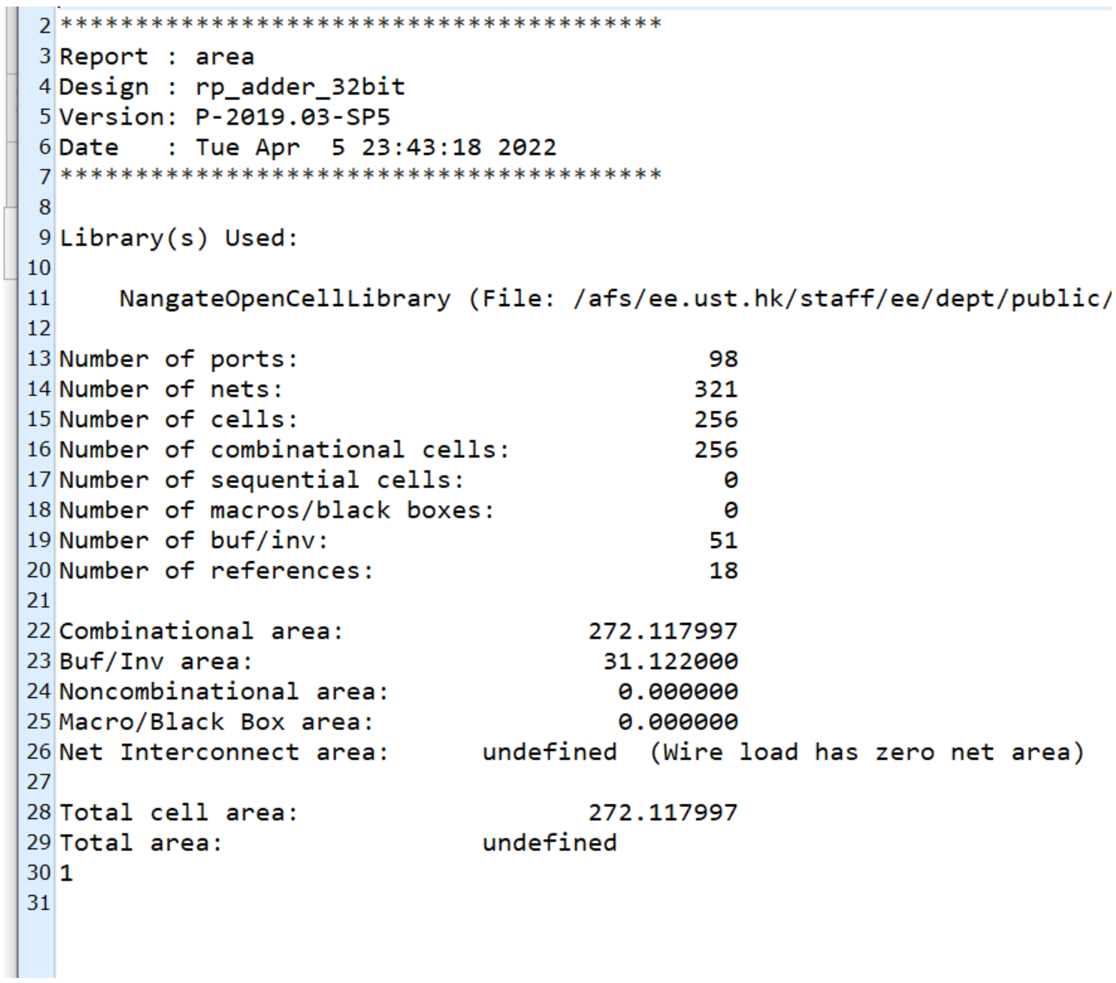


```
n339 (net)                      2                 0.00       2.74 r
U209/ZN (NAND2_X1)                      0.02      0.06       2.80 f
n263 (net)                      1                 0.00       2.80 f
U279/ZN (OAI211_X1)                     0.10      0.09       2.89 r
n205 (net)                      2                 0.00       2.89 r
U229/ZN (NAND2_X1)                      0.03      0.08       2.98 f
n351 (net)                      2                 0.00       2.98 f
U224/ZN (NAND2_X1)                      0.03      0.07       3.04 r
n266 (net)                      1                 0.00       3.04 r
U381/ZN (OAI211_X1)                     0.05      0.09       3.14 f
n360 (net)                      2                 0.00       3.14 f
U281/ZN (NAND2_X1)                      0.04      0.09       3.22 r
n367 (net)                      2                 0.00       3.22 r
U280/ZN (NAND2_X1)                      0.02      0.06       3.29 f
n267 (net)                      2                 0.00       3.29 f
U285/ZN (OAI211_X1)                     0.08      0.09       3.37 r
n372 (net)                      1                 0.00       3.37 r
U287/ZN (NAND2_X1)                      0.03      0.07       3.44 f
n379 (net)                      2                 0.00       3.44 f
U282/ZN (NAND2_X1)                      0.03      0.07       3.51 r
n270 (net)                      1                 0.00       3.51 r
U230/ZN (OAI211_X1)                     0.05      0.10       3.61 f
n174 (net)                      2                 0.00       3.61 f
U234/ZN (AND2_X2)                       0.02      0.13       3.74 f
n191 (net)                      2                 0.00       3.74 f
U232/ZN (OAI21_X2)                      0.06      0.11       3.85 r
n390 (net)                      1                 0.00       3.85 r
U261/ZN (XNOR2_X1)                      0.06      0.14       3.99 r
sum[30] (net)                   1                 0.00       3.99 r
sum[30] (out)                           0.06      0.01       4.00 r
data arrival time                                            4.00

clock VCLK (rise edge)                            5.00       5.00
clock network delay (ideal)                       0.00       5.00
output external delay                            -1.00       4.00
data required time                                           4.00
--------------------------------------------------------------------
data required time                                           4.00
data arrival time                                           -4.00
--------------------------------------------------------------------
slack (MET)                                                  0.00
```



```
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  14.1298 uW   (59%)
  Net Switching Power  =   9.7933 uW   (41%)
                         ---------
Total Dynamic Power    =  23.9231 uW  (100%)

Cell Leakage Power     =   4.3450 uW

Information: report_power power group summary does not include estimated clock tree power. (PWR-789)

                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000            0.0000  (   0.00%)
register           0.0000           0.0000            0.0000            0.0000  (   0.00%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational     14.1298           9.7933         4.3450e+03          28.2682  ( 100.00%)
--------------------------------------------------------------------------------------------------
Total             14.1298 uW        9.7933 uW      4.3450e+03 nW       28.2682 uW
1
```

<u>Area</u>



```
****************************************
Report : area
Design : rp_adder_32bit
Version: P-2019.03-SP5
Date   : Tue Apr  5 23:43:18 2022
****************************************

Library(s) Used:

    NangateOpenCellLibrary (File: /afs/ee.ust.hk/staff/ee/dept/public/

Number of ports:                           98
Number of nets:                           321
Number of cells:                          256
Number of combinational cells:            256
Number of sequential cells:                 0
Number of macros/black boxes:               0
Number of buf/inv:                         51
Number of references:                      18

Combinational area:                272.117997
Buf/Inv area:                       31.122000
Noncombinational area:               0.000000
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                   272.117997
Total area:                 undefined
1
```

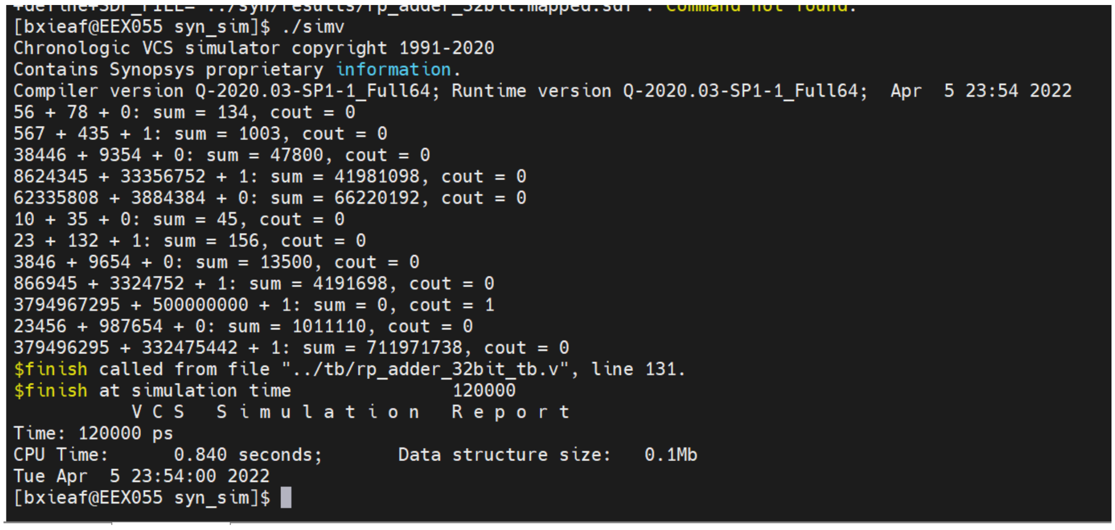
<u>Analysis:</u>

	Ripple carry adder	Brent-kung adder
power	28.3uW	28.6uW
area	272.12um2	252.2um2

Critical path delay	4ns	2.93ns

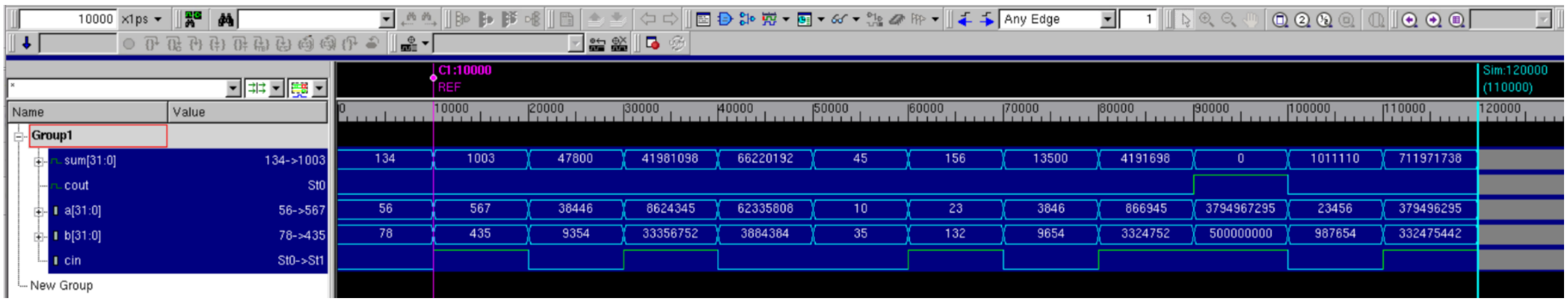
Comparing with the brent-kung adder, the total power consumption of ripple carry adder has about 0.3uW lower. However, the area is 20um2 larger than brent-kung adder. Also, the worst, the critical path delay of ripple carry is 4ns, which is larger than the delay of brent kung adder.

Synthesis simulation:

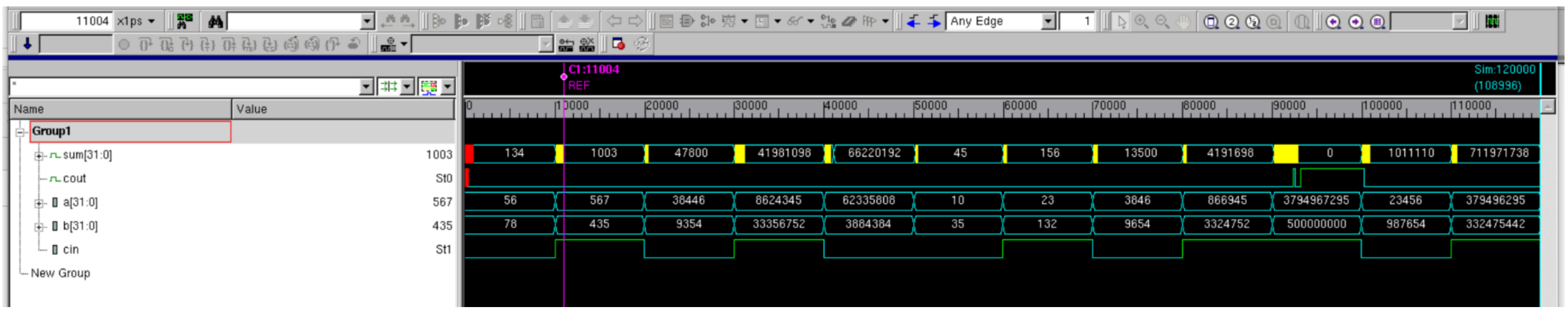


```
+define+SDF_FILE= ../syn/results/rp_adder_32bit.mapped.sdf : Command not found.
[bxieaf@EEX055 syn_sim]$ ./simv
Chronologic VCS simulator copyright 1991-2020
Contains Synopsys proprietary information.
Compiler version Q-2020.03-SP1-1_Full64; Runtime version Q-2020.03-SP1-1_Full64;  Apr  5 23:54 2022
56 + 78 + 0: sum = 134, cout = 0
567 + 435 + 1: sum = 1003, cout = 0
38446 + 9354 + 0: sum = 47800, cout = 0
8624345 + 33356752 + 1: sum = 41981098, cout = 0
62335808 + 3884384 + 0: sum = 66220192, cout = 0
10 + 35 + 0: sum = 45, cout = 0
23 + 132 + 1: sum = 156, cout = 0
3846 + 9654 + 0: sum = 13500, cout = 0
866945 + 3324752 + 1: sum = 4191698, cout = 0
3794967295 + 500000000 + 1: sum = 0, cout = 1
23456 + 987654 + 0: sum = 1011110, cout = 0
379496295 + 332475442 + 1: sum = 711971738, cout = 0
$finish called from file "../tb/rp_adder_32bit_tb.v", line 131.
$finish at simulation time                120000
           V C S   S i m u l a t i o n   R e p o r t
Time: 120000 ps
CPU Time:      0.840 seconds;       Data structure size:   0.1Mb
Tue Apr  5 23:54:00 2022
[bxieaf@EEX055 syn_sim]$ 
```

behavior simulation:

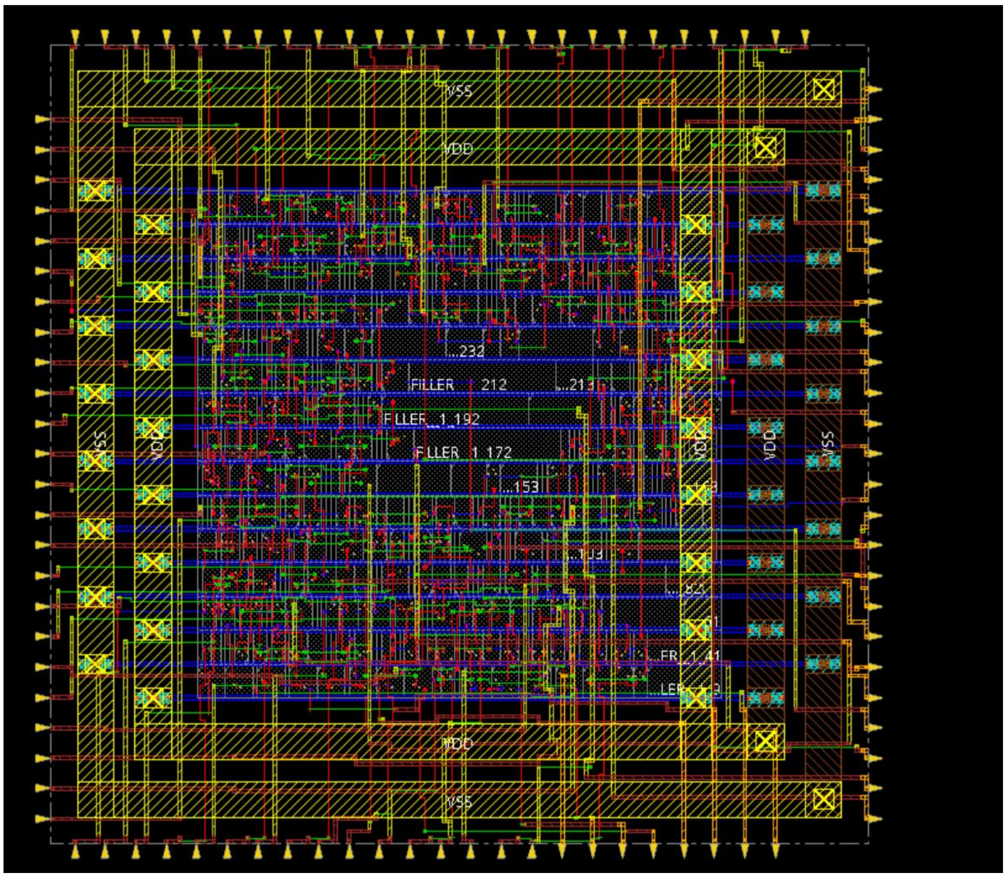


post synthesis simulation:

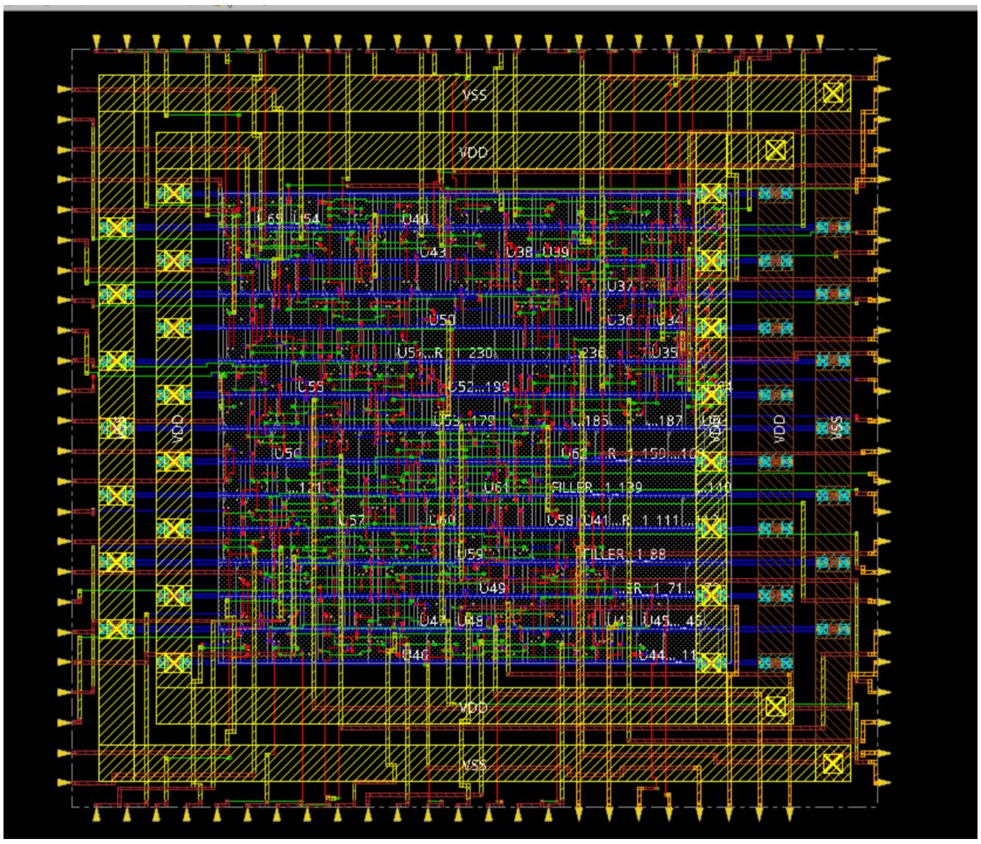


Layout

Ripple carry adder:

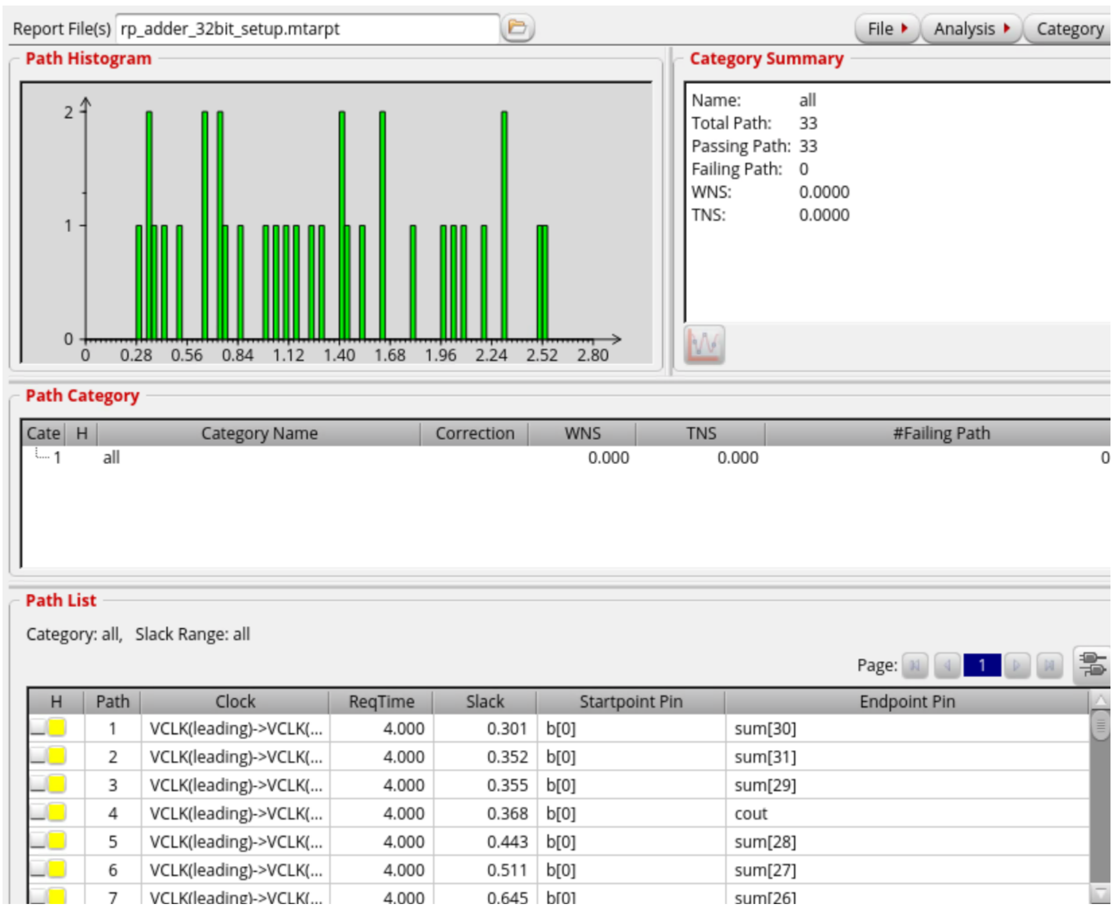


Brent-kung adder



Notice: Input delay(max,min)=(1ns,1ns), output delay(max, min)=(1ns,1ns)

Setup timing report (histogram) of ripple carry adder



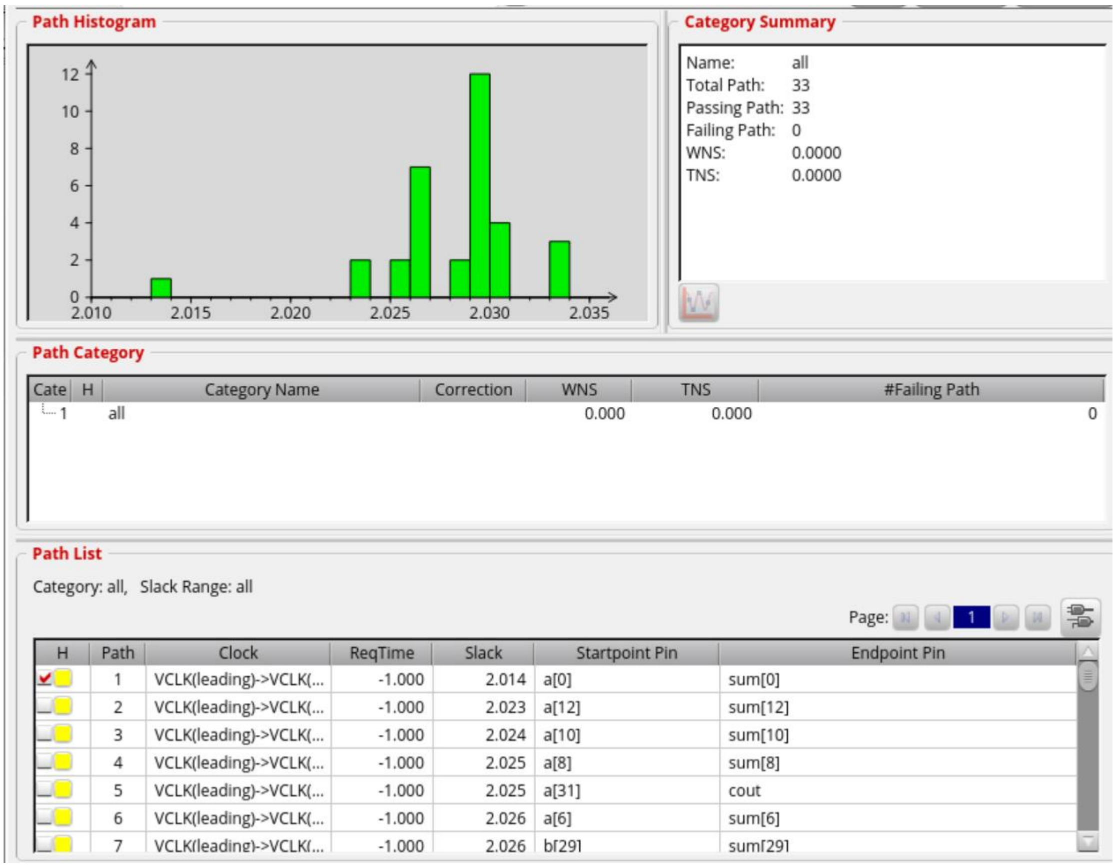
Name:	all
Total Path:	33
Passing Path:	33
Failing Path:	0
WNS:	0.0000
TNS:	0.0000

Cate	H	Category Name	Correction	WNS	TNS	#Failing Path
1		all		0.000	0.000	0

Category: all, Slack Range: all

H	Path	Clock	ReqTime	Slack	Startpoint Pin	Endpoint Pin
	1	VCLK(leading)->VCLK(...	4.000	0.301	b[0]	sum[30]
	2	VCLK(leading)->VCLK(...	4.000	0.352	b[0]	sum[31]
	3	VCLK(leading)->VCLK(...	4.000	0.355	b[0]	sum[29]
	4	VCLK(leading)->VCLK(...	4.000	0.368	b[0]	cout
	5	VCLK(leading)->VCLK(...	4.000	0.443	b[0]	sum[28]
	6	VCLK(leading)->VCLK(...	4.000	0.511	b[0]	sum[27]
	7	VCLK(leading)->VCLK(...	4.000	0.645	b[0]	sum[26]

- Hold timing report (histogram) of ripple carry adder



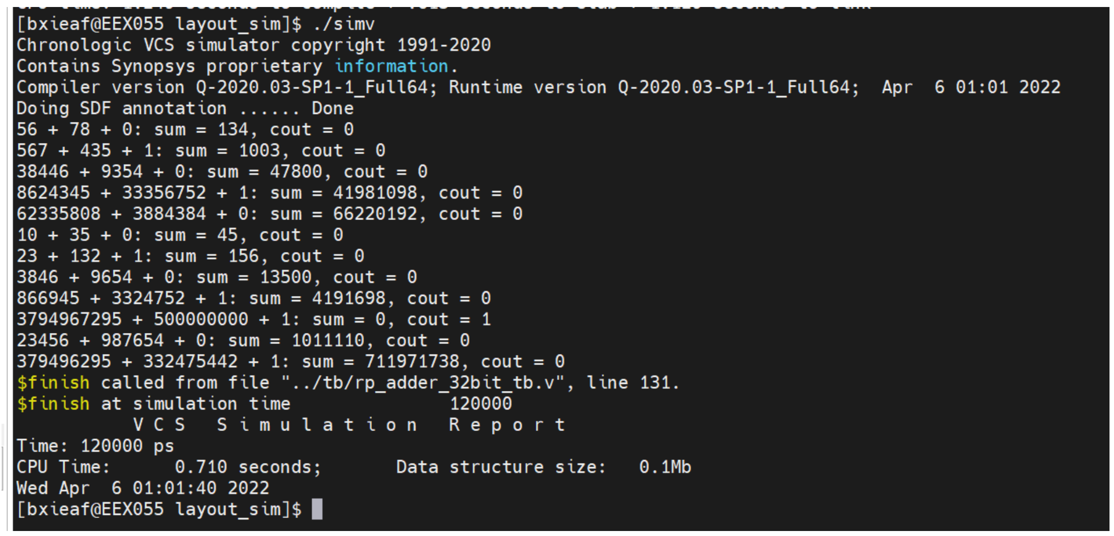
Name:	all
Total Path:	33
Passing Path:	33
Failing Path:	0
WNS:	0.0000
TNS:	0.0000

Cate	H	Category Name	Correction	WNS	TNS	#Failing Path
1		all		0.000	0.000	0

Category: all, Slack Range: all

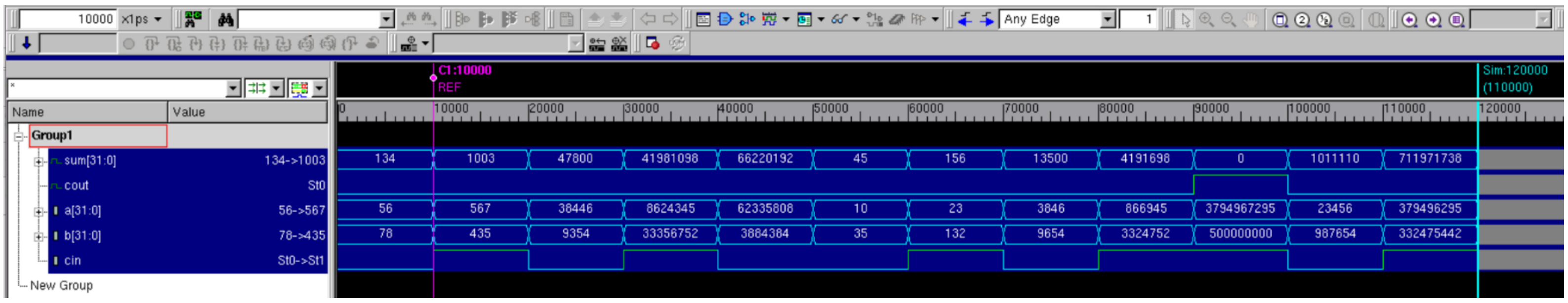
H	Path	Clock	ReqTime	Slack	Startpoint Pin	Endpoint Pin
✓	1	VCLK(leading)->VCLK(...	-1.000	2.014	a[0]	sum[0]
	2	VCLK(leading)->VCLK(...	-1.000	2.023	a[12]	sum[12]
	3	VCLK(leading)->VCLK(...	-1.000	2.024	a[10]	sum[10]
	4	VCLK(leading)->VCLK(...	-1.000	2.025	a[8]	sum[8]
	5	VCLK(leading)->VCLK(...	-1.000	2.025	a[31]	cout
	6	VCLK(leading)->VCLK(...	-1.000	2.026	a[6]	sum[6]
	7	VCLK(leading)->VCLK(...	-1.000	2.026	b[29]	sum[29]

Post layout simulation

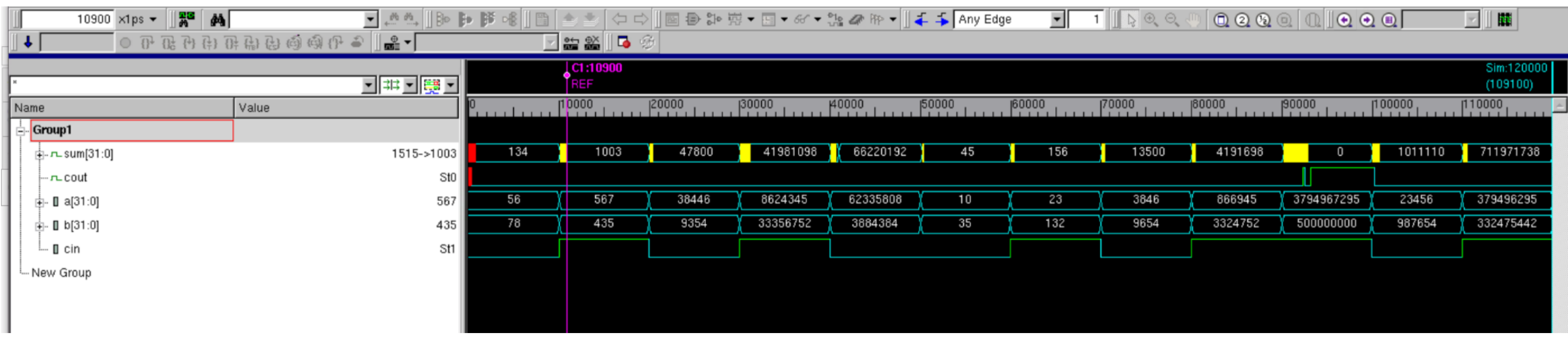


```
[bxieaf@EEX055 layout_sim]$ ./simv
Chronologic VCS simulator copyright 1991-2020
Contains Synopsys proprietary information.
Compiler version Q-2020.03-SP1-1_Full64; Runtime version Q-2020.03-SP1-1_Full64;  Apr  6 01:01 2022
Doing SDF annotation ...... Done
56 + 78 + 0: sum = 134, cout = 0
567 + 435 + 1: sum = 1003, cout = 0
38446 + 9354 + 0: sum = 47800, cout = 0
8624345 + 33356752 + 1: sum = 41981098, cout = 0
62335808 + 3884384 + 0: sum = 66220192, cout = 0
10 + 35 + 0: sum = 45, cout = 0
23 + 132 + 1: sum = 156, cout = 0
3846 + 9654 + 0: sum = 13500, cout = 0
866945 + 3324752 + 1: sum = 4191698, cout = 0
3794967295 + 500000000 + 1: sum = 0, cout = 1
23456 + 987654 + 0: sum = 1011110, cout = 0
379496295 + 332475442 + 1: sum = 711971738, cout = 0
$finish called from file "../tb/rp_adder_32bit_tb.v", line 131.
$finish at simulation time                120000
           V C S   S i m u l a t i o n   R e p o r t
Time: 120000 ps
CPU Time:      0.710 seconds;       Data structure size:   0.1Mb
Wed Apr  6 01:01:40 2022
[bxieaf@EEX055 layout_sim]$ 
```

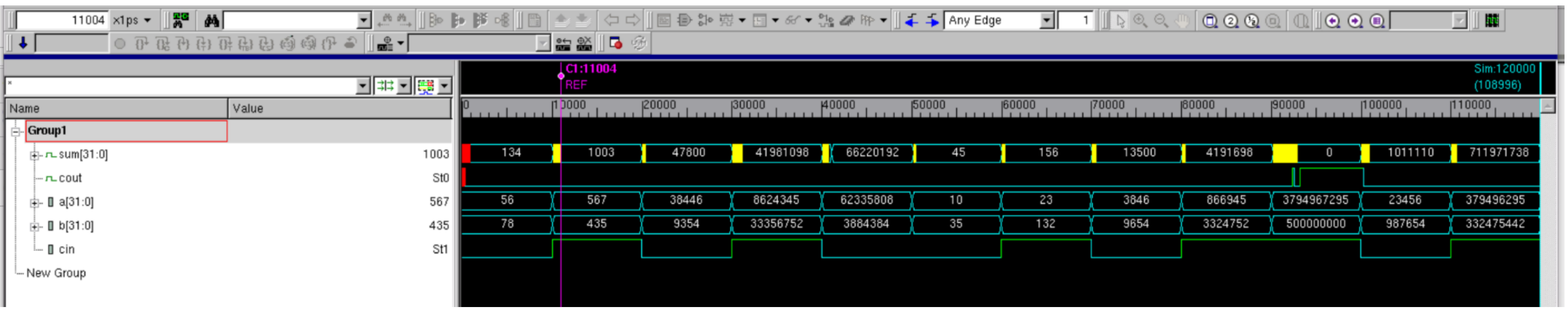
behavior simulation:



post synthesis simulation:



Post layout simulation:



C1:11004
REF
Sim:120000 (108996)
Name
Value
Group1
sum[31:0] 1003
cout St0
a[31:0] 567
b[31:0] 435
cin St1
New Group
134 1003 47800 41981098 66220192 45 156 13500 4191698 0 1011110 711971738
56 567 38446 8624345 62335808 10 23 3846 866945 3794967295 23456 379496295
78 435 9354 33356752 3884384 35 132 9654 3324752 500000000 987654 332475442