EEE 413 Introduction to VLSI Design - Laboratory Final Project

Full-Chip Design Project - 8-bit Synchronous Carry-Look-Ahead Adder

Members and Contribitures:

Ege Ereren

Barış Güzel

THIS PAGE LEFT BLANK INTENTIONALLY

TABLE OF CONTENTS

LIST OF FIGURES

1. OBJECTIVES

Objective of this lab is to achieve a fully functional 8- bit synchronous carry lookahead adder. Verilog design process with electric VLSI integration will be used to design this project. Full chip design criteria will be studied, and adder will be designed accordingly.

2. INTRODUCTION

Carry Lookahead Adder is a type of combination circuit that eliminates carry being propagated through the system. Only Cin is used to determine the output. There were multiple design options. First a generator circuit is designed with two xor gates and one and gate. This circuit generates P and G outputs along with the S output which is the output of the sum. Then each P and G of each bit is fed to the carry propagation circuit. This circuit is then divided into two parts for easier operation in Electric VLSI. Lastly as seen in the figure 4 below discussed levels are combined to create full CLA design. Register that is already achieved in previous labs are converted into Verilog codes and same operations are done as the CLA to create compact registers. 2 8 bit register and one Cin is combined to a 17bit input rising edge register. For output 8 bit result and Cout is combined to a 9 bit register. Both registers' delays are minimized by inverter chains both for clock and reset signals.

3. QUESTIONS

3.1 Question 1

3.1.1 Rising-Edge Triggered Synchronous 8-bit CLA Schematic

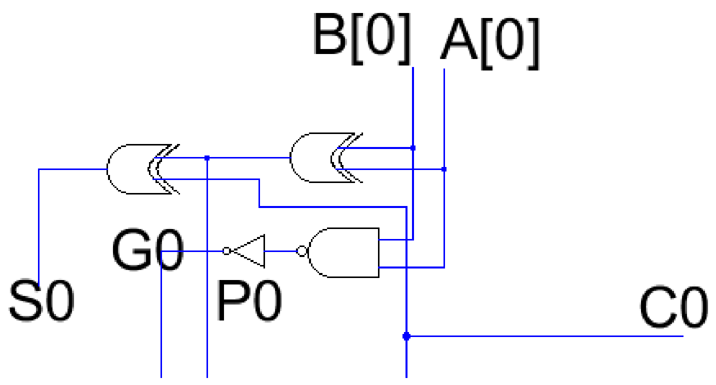


Figure 1 CLA Generator

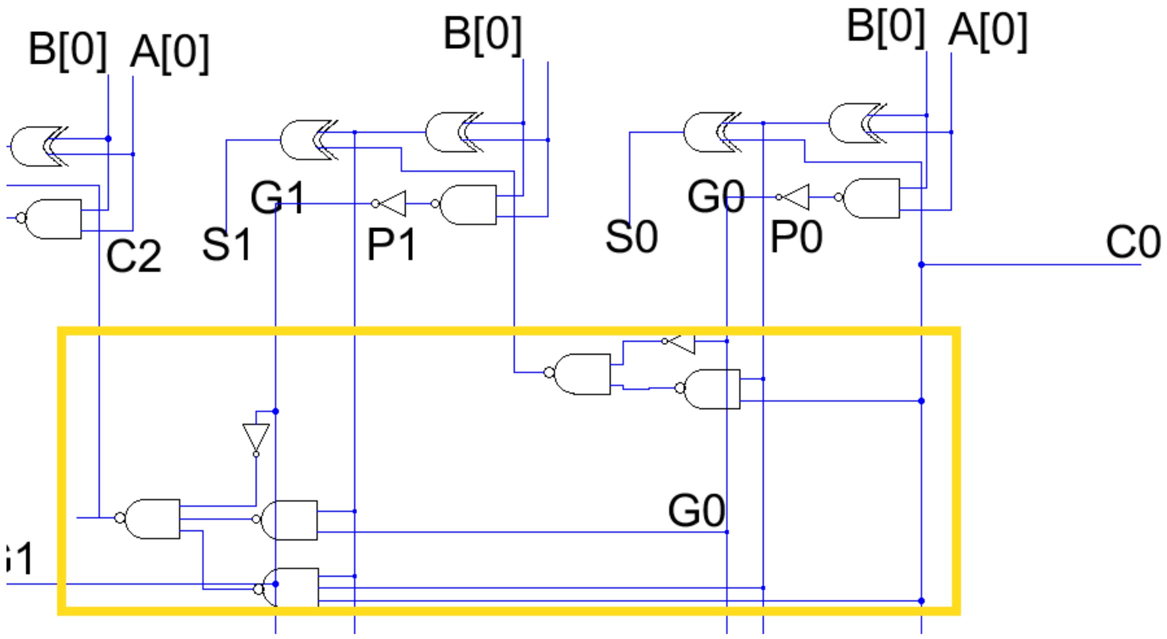


Figure 2 CLA Adder Level-1

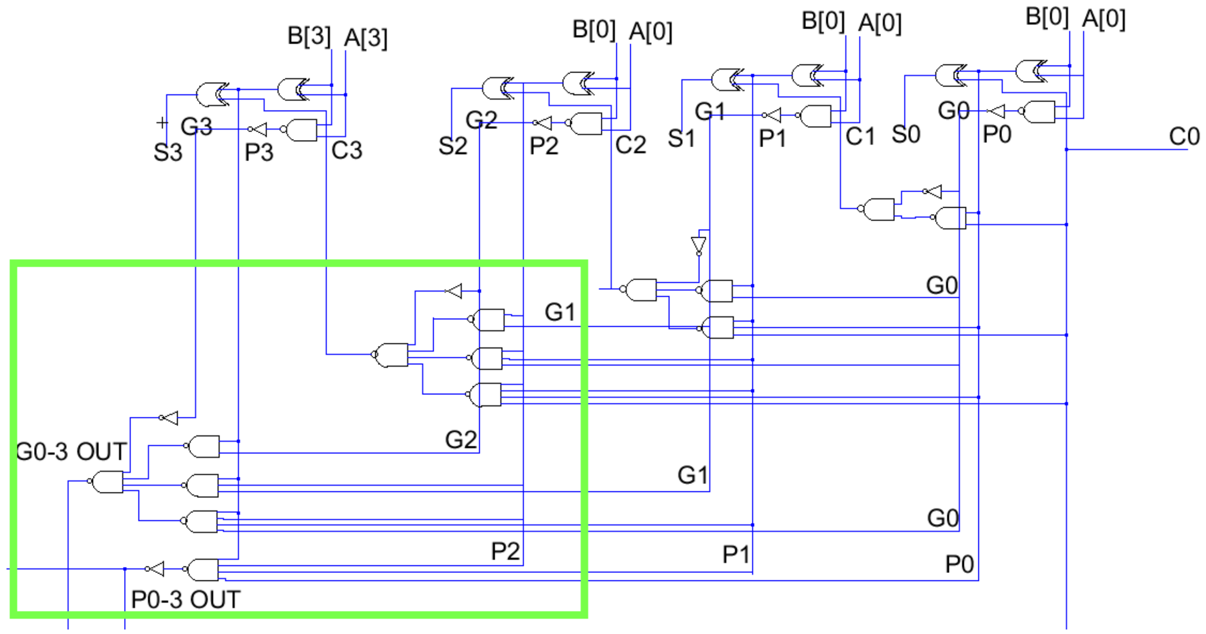


Figure 3 CLA Adder Level-2

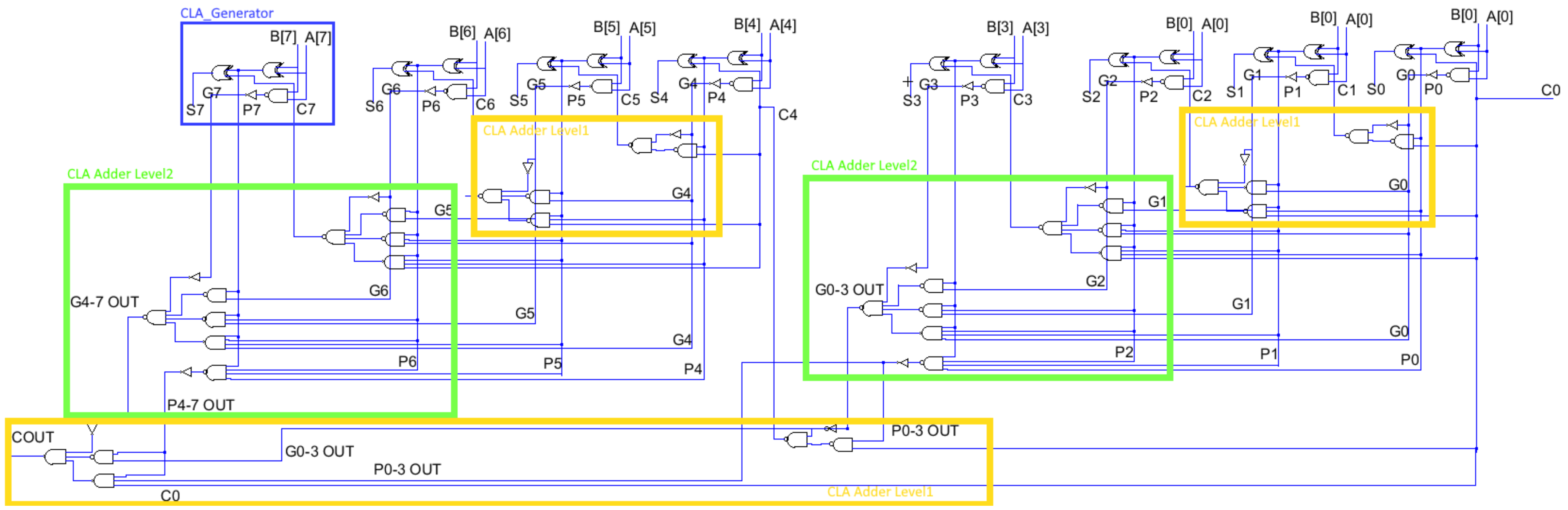


Figure 4 Full Circuit Design of CLA

3.1.2 Verilog Simulation

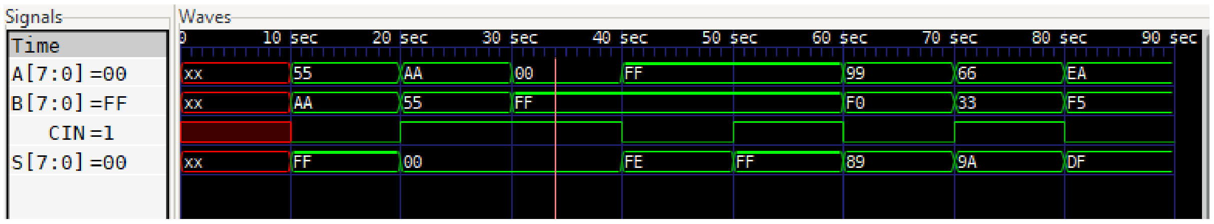
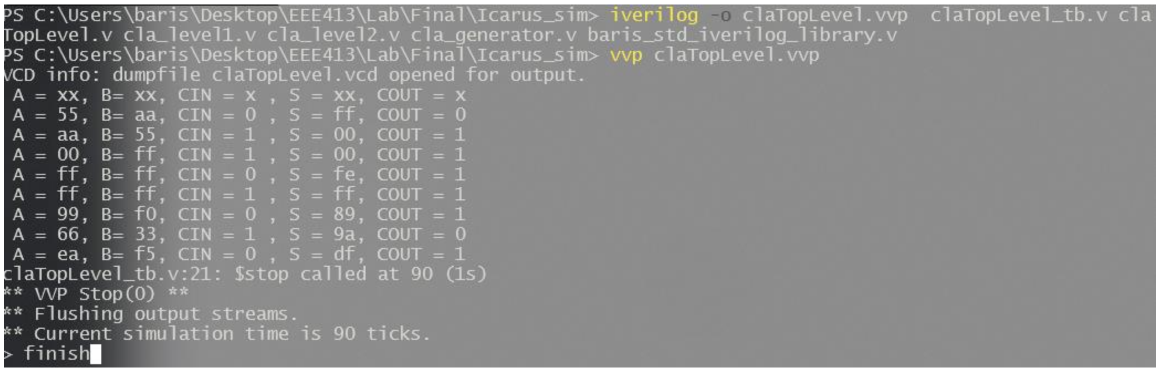
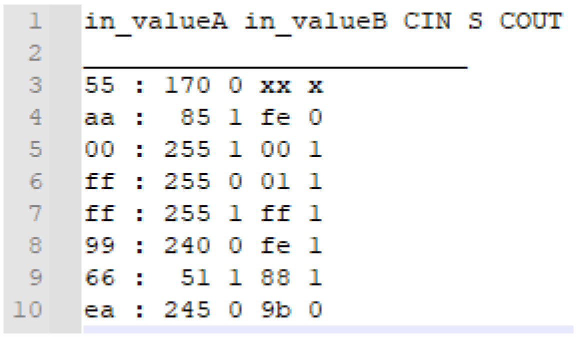


Figure 5 Gtkwave Simulation Screen Capture



```
PS C:\Users\baris\Desktop\EEE413\Lab\Final\Icarus_sim> iverilog -o claTopLevel.vvp  claTopLevel_tb.v cla
TopLevel.v cla_level1.v cla_level2.v cla_generator.v baris_std_iverilog_library.v
PS C:\Users\baris\Desktop\EEE413\Lab\Final\Icarus_sim> vvp claTopLevel.vvp
VCD info: dumpfile claTopLevel.vcd opened for output.
 A = xx, B= xx, CIN = x , S = xx, COUT = x
 A = 55, B= aa, CIN = 0 , S = ff, COUT = 0
 A = aa, B= 55, CIN = 1 , S = 00, COUT = 1
 A = 00, B= ff, CIN = 1 , S = 00, COUT = 1
 A = ff, B= ff, CIN = 0 , S = fe, COUT = 1
 A = ff, B= ff, CIN = 1 , S = ff, COUT = 1
 A = 99, B= f0, CIN = 0 , S = 89, COUT = 1
 A = 66, B= 33, CIN = 1 , S = 9a, COUT = 0
 A = ea, B= f5, CIN = 0 , S = df, COUT = 1
claTopLevel_tb.v:21: $stop called at 90 (1s)
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 90 ticks.
> finish
```

Figure 6 iverilog Terminal Output



```
in_valueA in_valueB CIN S COUT
________________________
55 : 170 0 xx x
aa :  85 1 fe 0
00 : 255 1 00 1
ff : 255 0 01 1
ff : 255 1 ff 1
99 : 240 0 fe 1
66 :  51 1 88 1
ea : 245 0 9b 0
```

Figure 7 output.dat File

3.2 Question 2

A custom standard cell is constructed according to the design’s needs. No nor gates are used in design. Pmos being bigger than nmos and since nor is a pmos heavy design there were little room for wiring. Whole design is summarized with mainly nand gates and inverters while total number of transistors used didn’t change. In the design discussed in the first question: NAND4, NAND3, NAND2, XOR and inverter is used. Similar technique that practiced in laboratory 3 is applied here when bigger sized inverter is needed. Inverters are connected in parallel to achieve desired size. All the design screen captures, and error checks are documented in appendix A.

3.3 Question 3

Optimized conceptual design is converted into layout design in Electric VLSI. Codes in the previous chapter are used to create each subcircuit layout. In the highest design level modules are aligned manually and wiring is done by the automated Electric VLSI tool. Automating the floorplan in bigger sized modules were less effective than doing it manually. Every subcircuit layout with simulations is shown in Appendix C.

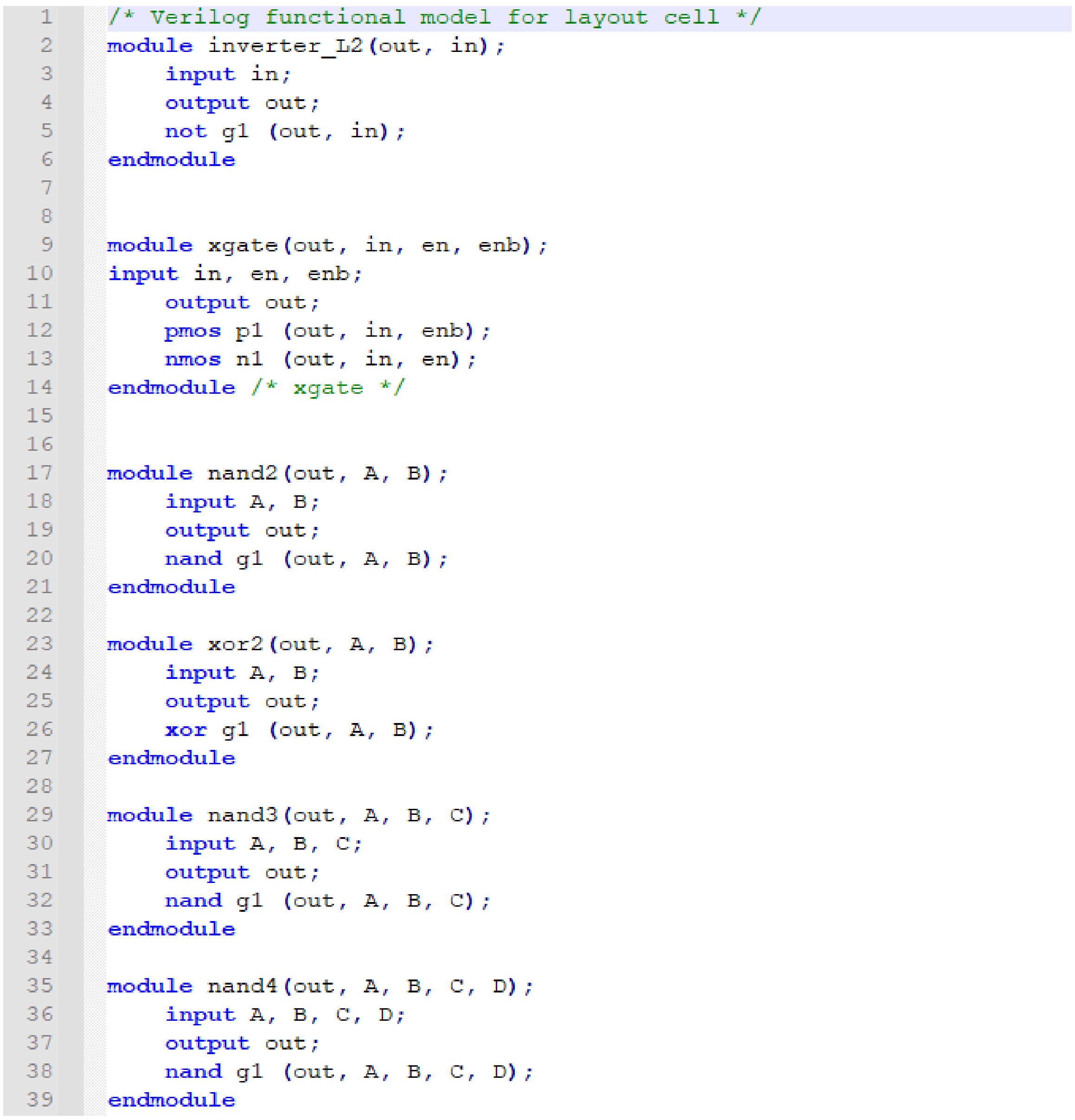
4. CONCLUSION

At first, we test our circuit with Verilog testbench tool. We stick to building our circuit with Verilog code. For some part Electric VLSI auto routing tool is better. However, Electric VLSI could not handle bigger circuits. Therefore, we divided our circuit. For example, memory and inverter chain build separately. We combined manually. Unfortunately, we could not achieve full circuit chip design in this lab. We were able to combine input registers and carry look ahead circuit. These two circuits can work separately. For some cases combined circuit has errors and required more further debugging to eliminate those errors.

5. REFFERENCES

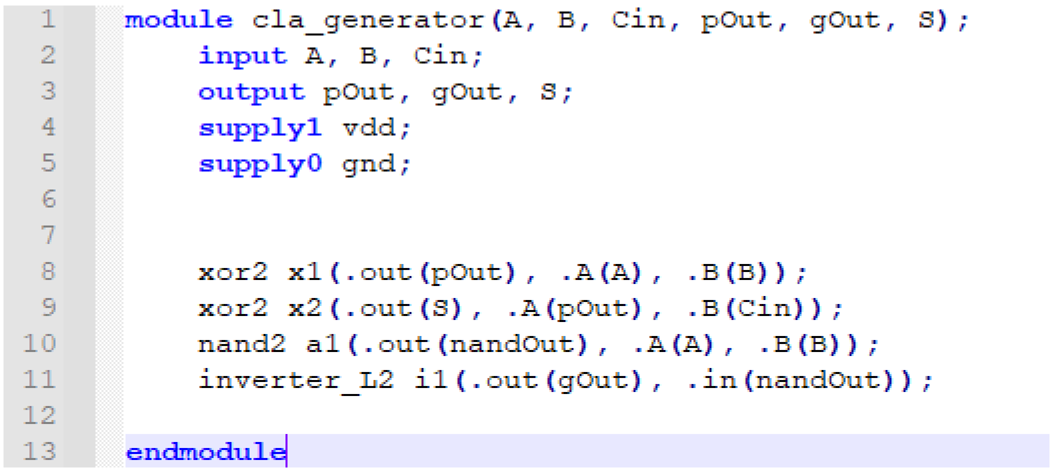
METU NCC. (2021). EEE 413 – Introduction to VLSI Design Lab Modules 4 & 5: Full-Chip Design Project -8-bit Synchronous Carry-Look-Ahead Adder.

APPENDIX A



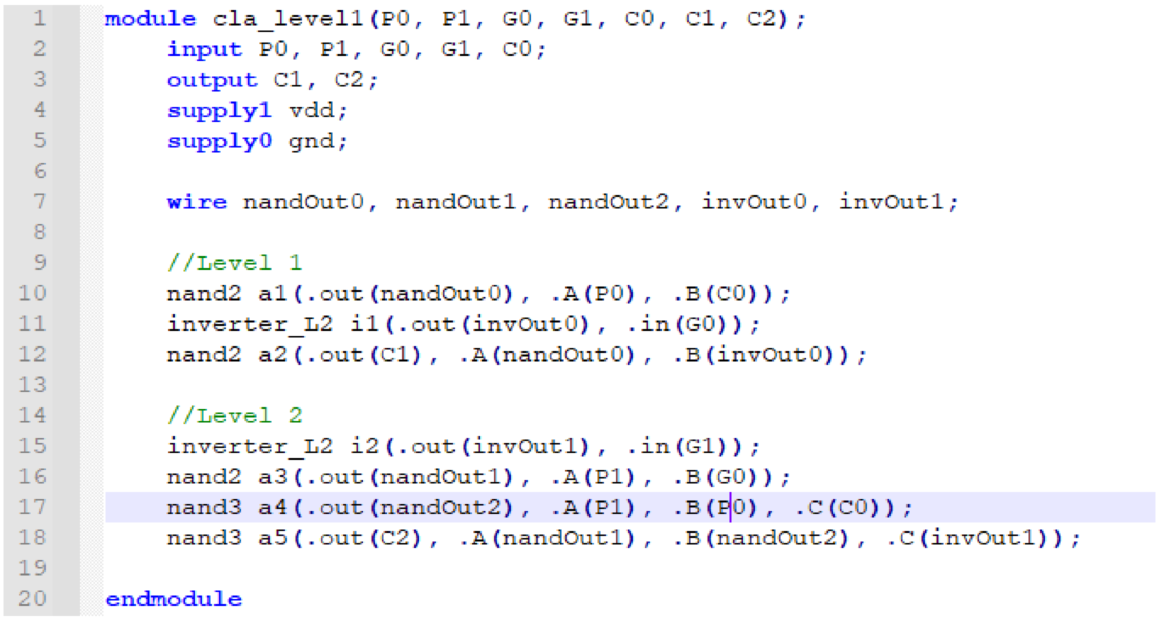
```
/* Verilog functional model for layout cell */
module inverter_L2(out, in);
    input in;
    output out;
    not g1 (out, in);
endmodule


module xgate(out, in, en, enb);
input in, en, enb;
    output out;
    pmos p1 (out, in, enb);
    nmos n1 (out, in, en);
endmodule /* xgate */


module nand2(out, A, B);
    input A, B;
    output out;
    nand g1 (out, A, B);
endmodule

module xor2(out, A, B);
    input A, B;
    output out;
    xor g1 (out, A, B);
endmodule

module nand3(out, A, B, C);
    input A, B, C;
    output out;
    nand g1 (out, A, B, C);
endmodule

module nand4(out, A, B, C, D);
    input A, B, C, D;
    output out;
    nand g1 (out, A, B, C, D);
endmodule
```

Figure 8 std_iverilog_library.v



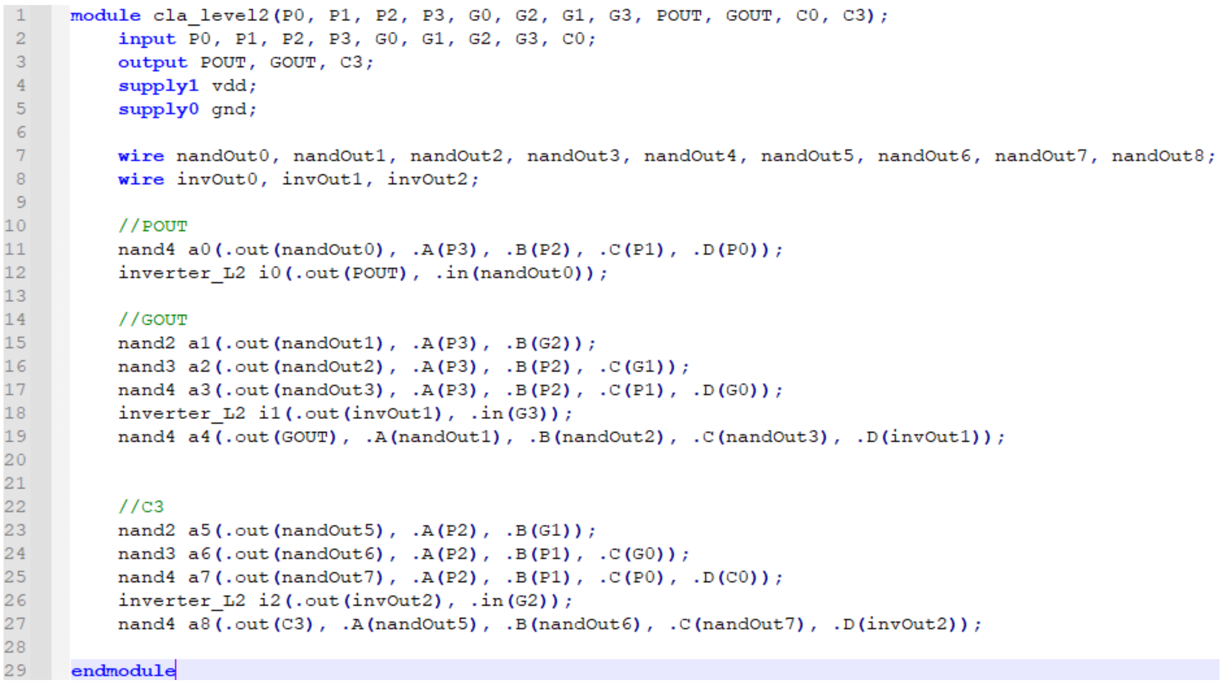
```
module cla_generator(A, B, Cin, pOut, gOut, S);
    input A, B, Cin;
    output pOut, gOut, S;
    supply1 vdd;
    supply0 gnd;


    xor2 x1(.out(pOut), .A(A), .B(B));
    xor2 x2(.out(S), .A(pOut), .B(Cin));
    nand2 a1(.out(nandOut), .A(A), .B(B));
    inverter_L2 i1(.out(gOut), .in(nandOut));

endmodule
```

Figure 9 cla_generator.v



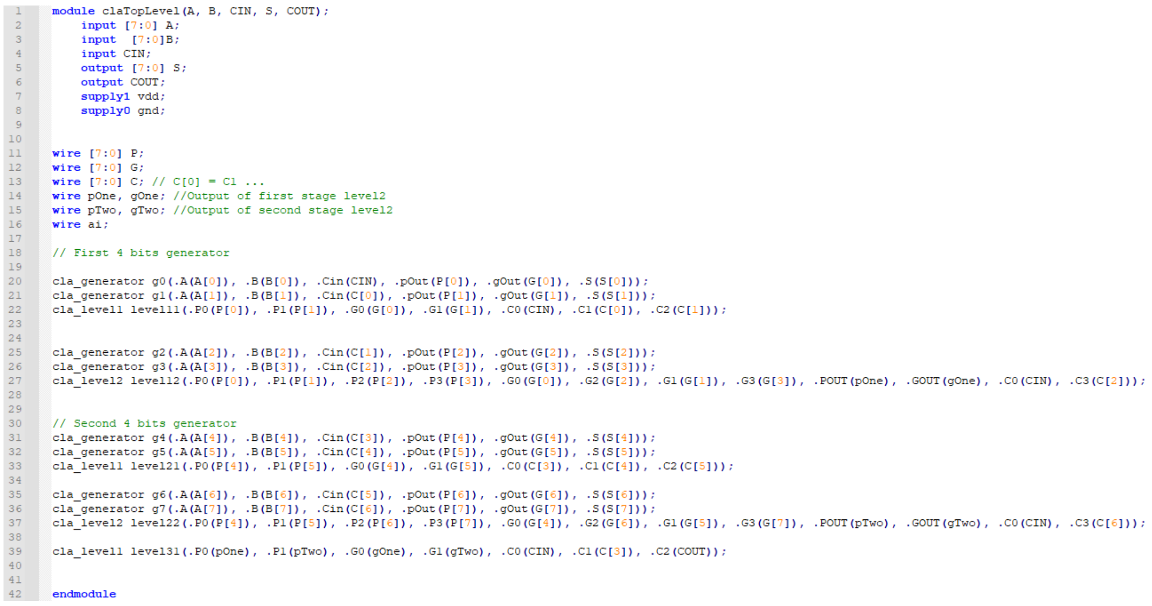
```
module cla_level1(P0, P1, G0, G1, C0, C1, C2);
    input P0, P1, G0, G1, C0;
    output C1, C2;
    supply1 vdd;
    supply0 gnd;

    wire nandOut0, nandOut1, nandOut2, invOut0, invOut1;

    //Level 1
    nand2 a1(.out(nandOut0), .A(P0), .B(C0));
    inverter_L2 i1(.out(invOut0), .in(G0));
    nand2 a2(.out(C1), .A(nandOut0), .B(invOut0));

    //Level 2
    inverter_L2 i2(.out(invOut1), .in(G1));
    nand2 a3(.out(nandOut1), .A(P1), .B(G0));
    nand3 a4(.out(nandOut2), .A(P1), .B(P0), .C(C0));
    nand3 a5(.out(C2), .A(nandOut1), .B(nandOut2), .C(invOut1));

endmodule
```

Figure 10 cla_level1.v



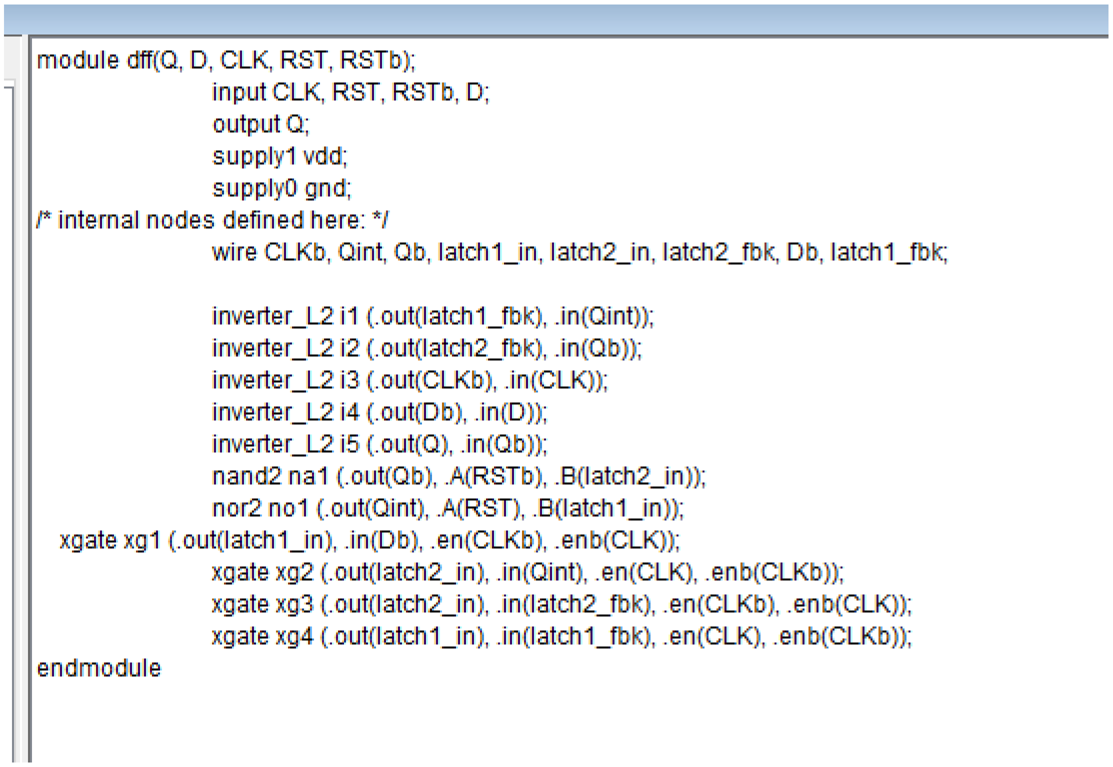
```verilog
module cla_level2(P0, P1, P2, P3, G0, G2, G1, G3, POUT, GOUT, C0, C3);
    input P0, P1, P2, P3, G0, G1, G2, G3, C0;
    output POUT, GOUT, C3;
    supply1 vdd;
    supply0 gnd;

    wire nandOut0, nandOut1, nandOut2, nandOut3, nandOut4, nandOut5, nandOut6, nandOut7, nandOut8;
    wire invOut0, invOut1, invOut2;

    //POUT
    nand4 a0(.out(nandOut0), .A(P3), .B(P2), .C(P1), .D(P0));
    inverter_L2 i0(.out(POUT), .in(nandOut0));

    //GOUT
    nand2 a1(.out(nandOut1), .A(P3), .B(G2));
    nand3 a2(.out(nandOut2), .A(P3), .B(P2), .C(G1));
    nand4 a3(.out(nandOut3), .A(P3), .B(P2), .C(P1), .D(G0));
    inverter_L2 i1(.out(invOut1), .in(G3));
    nand4 a4(.out(GOUT), .A(nandOut1), .B(nandOut2), .C(nandOut3), .D(invOut1));


    //C3
    nand2 a5(.out(nandOut5), .A(P2), .B(G1));
    nand3 a6(.out(nandOut6), .A(P2), .B(P1), .C(G0));
    nand4 a7(.out(nandOut7), .A(P2), .B(P1), .C(P0), .D(C0));
    inverter_L2 i2(.out(invOut2), .in(G2));
    nand4 a8(.out(C3), .A(nandOut5), .B(nandOut6), .C(nandOut7), .D(invOut2));

endmodule
```

Figure 11 cla_level2.v



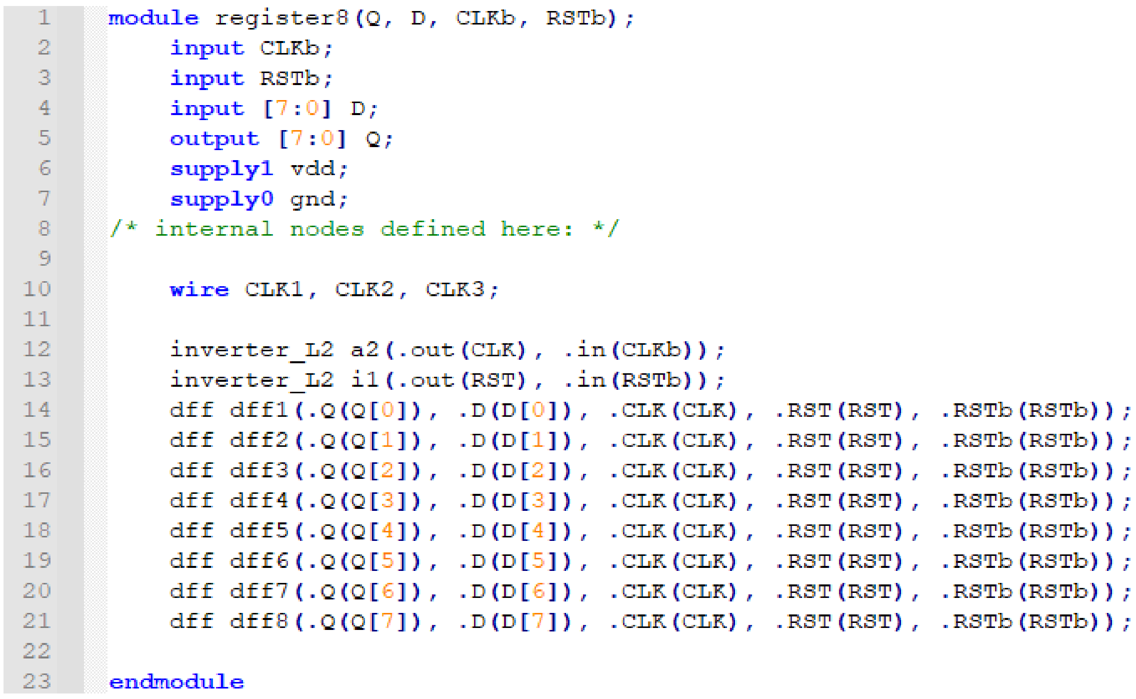
```verilog
module claTopLevel(A, B, CIN, S, COUT);
    input [7:0] A;
    input  [7:0]B;
    input CIN;
    output [7:0] S;
    output COUT;
    supply1 vdd;
    supply0 gnd;


wire [7:0] P;
wire [7:0] G;
wire [7:0] C; // C[0] = C1 ...
wire pOne, gOne; //Output of first stage level2
wire pTwo, gTwo; //Output of second stage level2
wire ai;

// First 4 bits generator

cla_generator g0(.A(A[0]), .B(B[0]), .Cin(CIN), .pOut(P[0]), .gOut(G[0]), .S(S[0]));
cla_generator g1(.A(A[1]), .B(B[1]), .Cin(C[0]), .pOut(P[1]), .gOut(G[1]), .S(S[1]));
cla_level1 level11(.P0(P[0]), .P1(P[1]), .G0(G[0]), .G1(G[1]), .C0(CIN), .C1(C[0]), .C2(C[1]));


cla_generator g2(.A(A[2]), .B(B[2]), .Cin(C[1]), .pOut(P[2]), .gOut(G[2]), .S(S[2]));
cla_generator g3(.A(A[3]), .B(B[3]), .Cin(C[2]), .pOut(P[3]), .gOut(G[3]), .S(S[3]));
cla_level2 level12(.P0(P[0]), .P1(P[1]), .P2(P[2]), .P3(P[3]), .G0(G[0]), .G2(G[2]), .G1(G[1]), .G3(G[3]), .POUT(pOne), .GOUT(gOne), .C0(CIN), .C3(C[2]));


// Second 4 bits generator
cla_generator g4(.A(A[4]), .B(B[4]), .Cin(C[3]), .pOut(P[4]), .gOut(G[4]), .S(S[4]));
cla_generator g5(.A(A[5]), .B(B[5]), .Cin(C[4]), .pOut(P[5]), .gOut(G[5]), .S(S[5]));
cla_level1 level21(.P0(P[4]), .P1(P[5]), .G0(G[4]), .G1(G[5]), .C0(C[3]), .C1(C[4]), .C2(C[5]));

cla_generator g6(.A(A[6]), .B(B[6]), .Cin(C[5]), .pOut(P[6]), .gOut(G[6]), .S(S[6]));
cla_generator g7(.A(A[7]), .B(B[7]), .Cin(C[6]), .pOut(P[7]), .gOut(G[7]), .S(S[7]));
cla_level2 level22(.P0(P[4]), .P1(P[5]), .P2(P[6]), .P3(P[7]), .G0(G[4]), .G2(G[6]), .G1(G[5]), .G3(G[7]), .POUT(pTwo), .GOUT(gTwo), .C0(CIN), .C3(C[6]));

cla_level1 level31(.P0(pOne), .P1(pTwo), .G0(gOne), .G1(gTwo), .C0(CIN), .C1(C[3]), .C2(COUT));


endmodule
```

Figure 12 claTopLevel.v



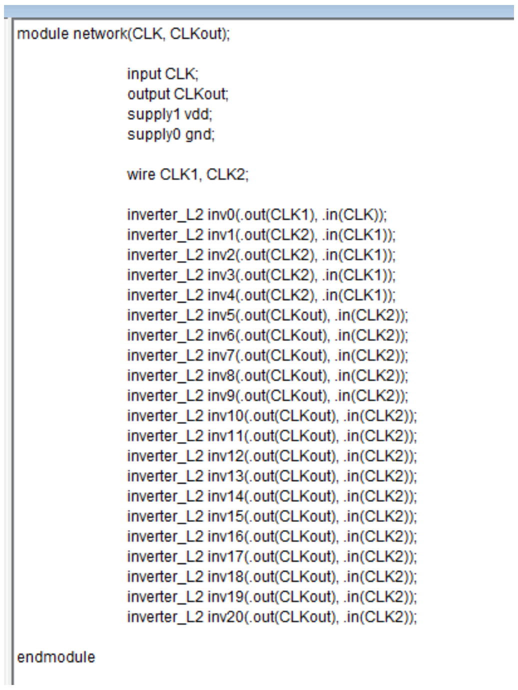
```
module dff(Q, D, CLK, RST, RSTb);
                input CLK, RST, RSTb, D;
                output Q;
                supply1 vdd;
                supply0 gnd;
/* internal nodes defined here: */
                wire CLKb, Qint, Qb, latch1_in, latch2_in, latch2_fbk, Db, latch1_fbk;

                inverter_L2 i1 (.out(latch1_fbk), .in(Qint));
                inverter_L2 i2 (.out(latch2_fbk), .in(Qb));
                inverter_L2 i3 (.out(CLKb), .in(CLK));
                inverter_L2 i4 (.out(Db), .in(D));
                inverter_L2 i5 (.out(Q), .in(Qb));
                nand2 na1 (.out(Qb), .A(RSTb), .B(latch2_in));
                nor2 no1 (.out(Qint), .A(RST), .B(latch1_in));
  xgate xg1 (.out(latch1_in), .in(Db), .en(CLKb), .enb(CLK));
                xgate xg2 (.out(latch2_in), .in(Qint), .en(CLK), .enb(CLKb));
                xgate xg3 (.out(latch2_in), .in(latch2_fbk), .en(CLKb), .enb(CLK));
                xgate xg4 (.out(latch1_in), .in(latch1_fbk), .en(CLK), .enb(CLKb));
endmodule
```

Figure 13 dff.v



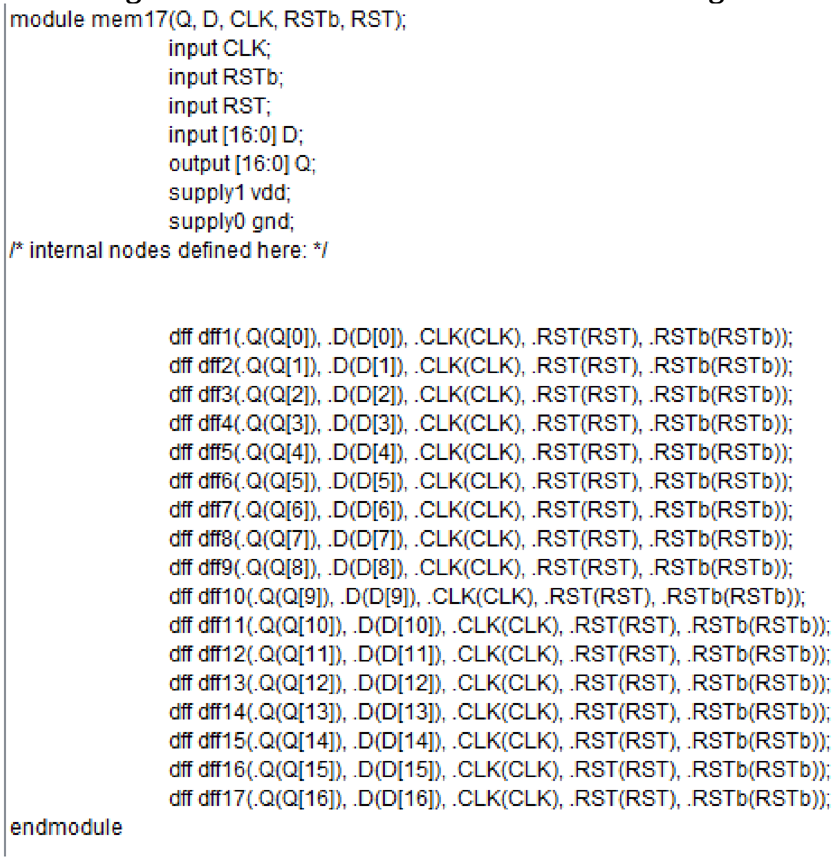
```
module register8(Q, D, CLKb, RSTb);
    input CLKb;
    input RSTb;
    input [7:0] D;
    output [7:0] Q;
    supply1 vdd;
    supply0 gnd;
/* internal nodes defined here: */

    wire CLK1, CLK2, CLK3;

    inverter_L2 a2(.out(CLK), .in(CLKb));
    inverter_L2 i1(.out(RST), .in(RSTb));
    dff dff1(.Q(Q[0]), .D(D[0]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
    dff dff2(.Q(Q[1]), .D(D[1]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
    dff dff3(.Q(Q[2]), .D(D[2]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
    dff dff4(.Q(Q[3]), .D(D[3]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
    dff dff5(.Q(Q[4]), .D(D[4]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
    dff dff6(.Q(Q[5]), .D(D[5]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
    dff dff7(.Q(Q[6]), .D(D[6]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
    dff dff8(.Q(Q[7]), .D(D[7]), .CLK(CLK), .RST(RST), .RSTb(RSTb));

endmodule
```

Figure 14 register8.v



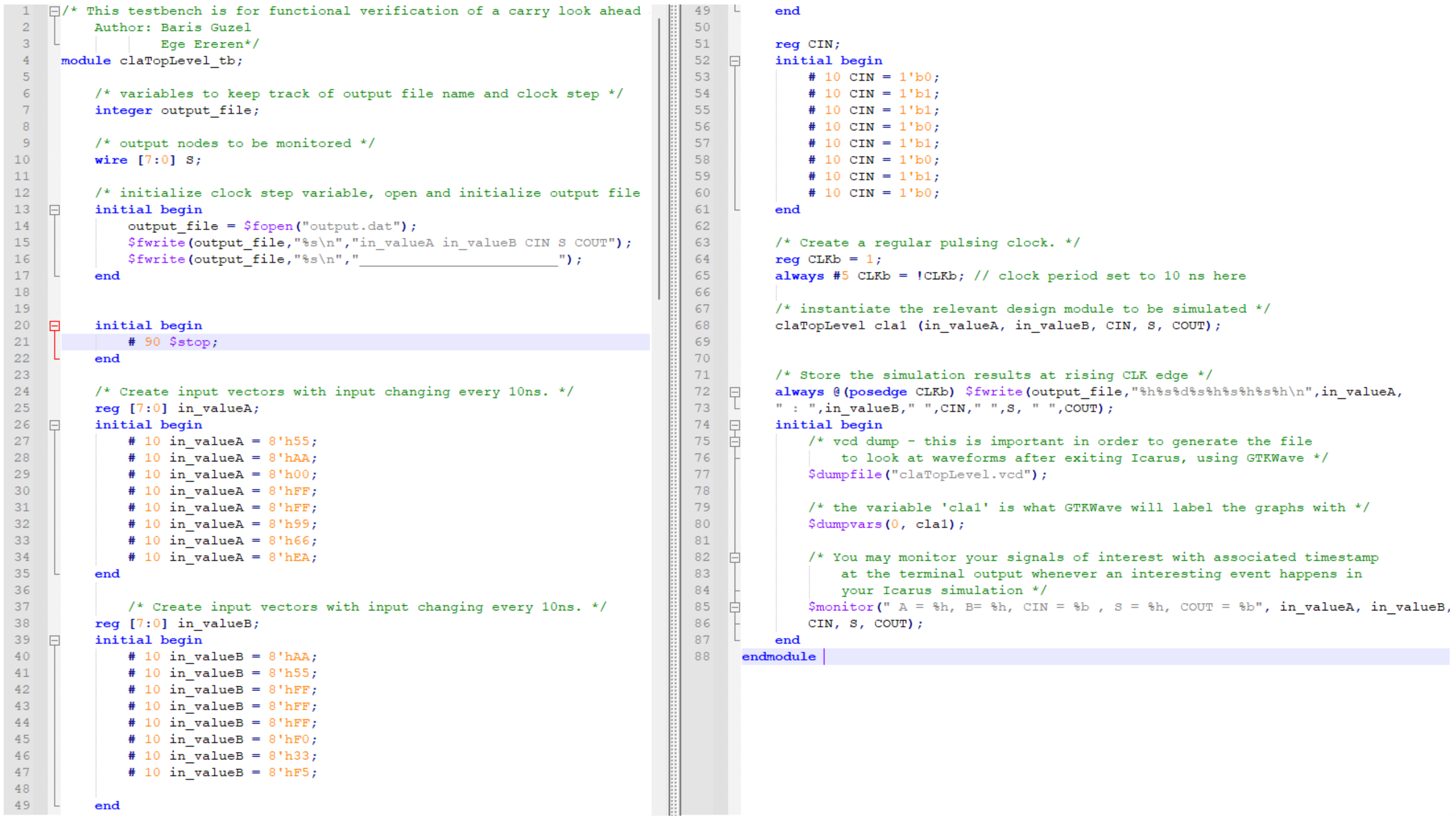
```
module network(CLK, CLKout);

                input CLK;
                output CLKout;
                supply1 vdd;
                supply0 gnd;

                wire CLK1, CLK2;

                inverter_L2 inv0(.out(CLK1), .in(CLK));
                inverter_L2 inv1(.out(CLK2), .in(CLK1));
                inverter_L2 inv2(.out(CLK2), .in(CLK1));
                inverter_L2 inv3(.out(CLK2), .in(CLK1));
                inverter_L2 inv4(.out(CLK2), .in(CLK1));
                inverter_L2 inv5(.out(CLKout), .in(CLK2));
                inverter_L2 inv6(.out(CLKout), .in(CLK2));
                inverter_L2 inv7(.out(CLKout), .in(CLK2));
                inverter_L2 inv8(.out(CLKout), .in(CLK2));
                inverter_L2 inv9(.out(CLKout), .in(CLK2));
                inverter_L2 inv10(.out(CLKout), .in(CLK2));
                inverter_L2 inv11(.out(CLKout), .in(CLK2));
                inverter_L2 inv12(.out(CLKout), .in(CLK2));
                inverter_L2 inv13(.out(CLKout), .in(CLK2));
                inverter_L2 inv14(.out(CLKout), .in(CLK2));
                inverter_L2 inv15(.out(CLKout), .in(CLK2));
                inverter_L2 inv16(.out(CLKout), .in(CLK2));
                inverter_L2 inv17(.out(CLKout), .in(CLK2));
                inverter_L2 inv18(.out(CLKout), .in(CLK2));
                inverter_L2 inv19(.out(CLKout), .in(CLK2));
                inverter_L2 inv20(.out(CLKout), .in(CLK2));

endmodule
```

Figure 15 Clock Inverter Chain for 17 bit Register



```
module mem17(Q, D, CLK, RSTb, RST);
                input CLK;
                input RSTb;
                input RST;
                input [16:0] D;
                output [16:0] Q;
                supply1 vdd;
                supply0 gnd;
/* internal nodes defined here: */


                dff dff1(.Q(Q[0]), .D(D[0]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff2(.Q(Q[1]), .D(D[1]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff3(.Q(Q[2]), .D(D[2]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff4(.Q(Q[3]), .D(D[3]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff5(.Q(Q[4]), .D(D[4]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff6(.Q(Q[5]), .D(D[5]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff7(.Q(Q[6]), .D(D[6]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff8(.Q(Q[7]), .D(D[7]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff9(.Q(Q[8]), .D(D[8]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff10(.Q(Q[9]), .D(D[9]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff11(.Q(Q[10]), .D(D[10]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff12(.Q(Q[11]), .D(D[11]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff13(.Q(Q[12]), .D(D[12]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff14(.Q(Q[13]), .D(D[13]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff15(.Q(Q[14]), .D(D[14]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff16(.Q(Q[15]), .D(D[15]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
                dff dff17(.Q(Q[16]), .D(D[16]), .CLK(CLK), .RST(RST), .RSTb(RSTb));
endmodule
```

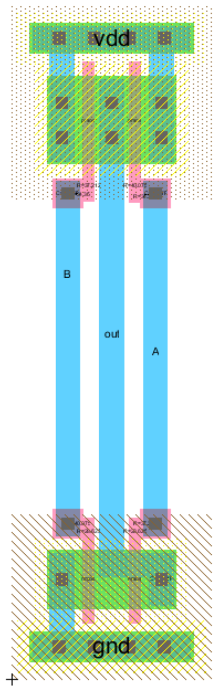
Figure 16 register17.v

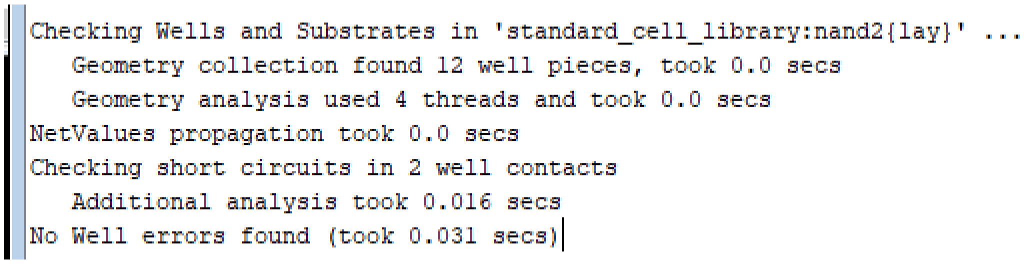


```verilog
/* This testbench is for functional verification of a carry look ahead
    Author: Baris Guzel
            Ege Ereren*/
module claTopLevel_tb;

    /* variables to keep track of output file name and clock step */
    integer output_file;

    /* output nodes to be monitored */
    wire [7:0] S;

    /* initialize clock step variable, open and initialize output file
    initial begin
        output_file = $fopen("output.dat");
        $fwrite(output_file,"%s\n","in_valueA in_valueB CIN S COUT");
        $fwrite(output_file,"%s\n","________________________");
    end


    initial begin
        # 90 $stop;
    end

    /* Create input vectors with input changing every 10ns. */
    reg [7:0] in_valueA;
    initial begin
        # 10 in_valueA = 8'h55;
        # 10 in_valueA = 8'hAA;
        # 10 in_valueA = 8'h00;
        # 10 in_valueA = 8'hFF;
        # 10 in_valueA = 8'hFF;
        # 10 in_valueA = 8'h99;
        # 10 in_valueA = 8'h66;
        # 10 in_valueA = 8'hEA;
    end

        /* Create input vectors with input changing every 10ns. */
    reg [7:0] in_valueB;
    initial begin
        # 10 in_valueB = 8'hAA;
        # 10 in_valueB = 8'h55;
        # 10 in_valueB = 8'hFF;
        # 10 in_valueB = 8'hFF;
        # 10 in_valueB = 8'hFF;
        # 10 in_valueB = 8'hF0;
        # 10 in_valueB = 8'h33;
        # 10 in_valueB = 8'hF5;

    end

    reg CIN;
    initial begin
        # 10 CIN = 1'b0;
        # 10 CIN = 1'b1;
        # 10 CIN = 1'b1;
        # 10 CIN = 1'b0;
        # 10 CIN = 1'b1;
        # 10 CIN = 1'b0;
        # 10 CIN = 1'b1;
        # 10 CIN = 1'b0;
    end

    /* Create a regular pulsing clock. */
    reg CLKb = 1;
    always #5 CLKb = !CLKb; // clock period set to 10 ns here

    /* instantiate the relevant design module to be simulated */
    claTopLevel cla1 (in_valueA, in_valueB, CIN, S, COUT);


    /* Store the simulation results at rising CLK edge */
    always @(posedge CLKb) $fwrite(output_file,"%h%s%d%s%h%s%h%s%h\n",in_valueA,
    " : ",in_valueB," ",CIN," ",S, " ",COUT);
    initial begin
        /* vcd dump - this is important in order to generate the file
            to look at waveforms after exiting Icarus, using GTKWave */
        $dumpfile("claTopLevel.vcd");

        /* the variable 'cla1' is what GTKWave will label the graphs with */
        $dumpvars(0, cla1);

        /* You may monitor your signals of interest with associated timestamp
            at the terminal output whenever an interesting event happens in
            your Icarus simulation */
        $monitor(" A = %h, B= %h, CIN = %b , S = %h, COUT = %b", in_valueA, in_valueB,
        CIN, S, COUT);
    end
endmodule
```

Figure 17 claTopLevel_tb.

APPENDIX B





```
Checking Wells and Substrates in 'standard_cell_library:nand2{lay}' ...
   Geometry collection found 12 well pieces, took 0.0 secs
   Geometry analysis used 4 threads and took 0.0 secs
NetValues propagation took 0.0 secs
Checking short circuits in 2 well contacts
   Additional analysis took 0.016 secs
No Well errors found (took 0.031 secs)
```

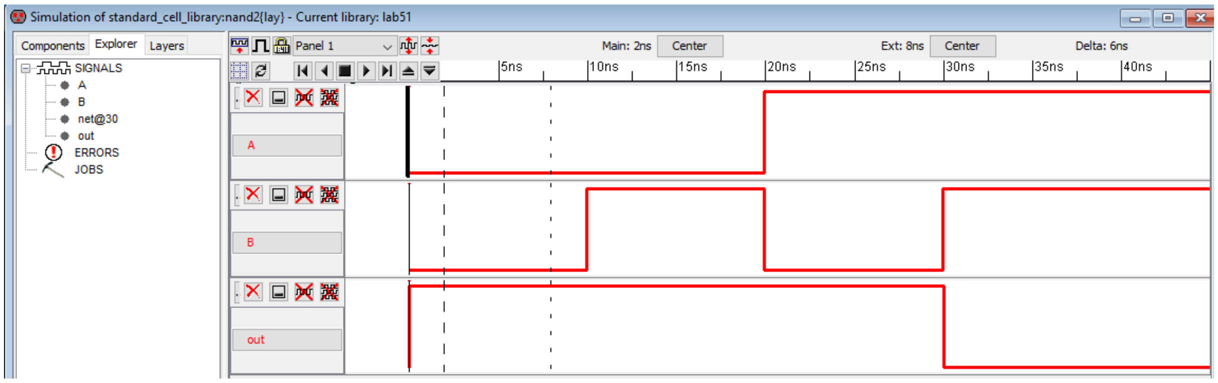
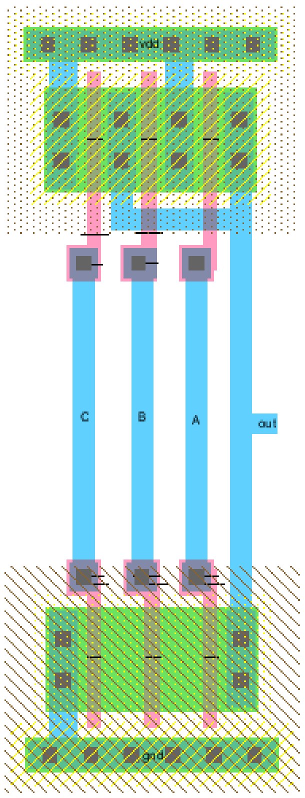
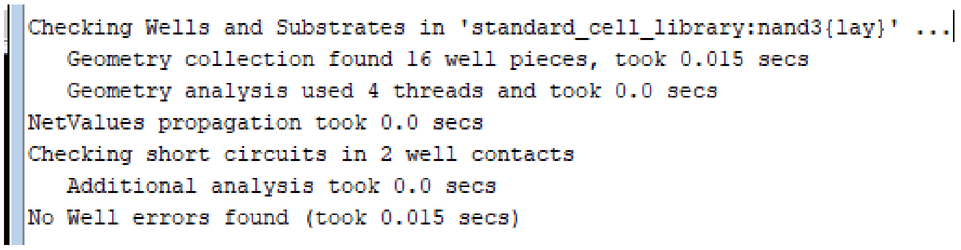


Figure 18 NAND2 Gate Layout Error Panel and Simulation Verification





```
Checking Wells and Substrates in 'standard_cell_library:nand3{lay}' ...
   Geometry collection found 16 well pieces, took 0.015 secs
   Geometry analysis used 4 threads and took 0.0 secs
NetValues propagation took 0.0 secs
Checking short circuits in 2 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.015 secs)
```

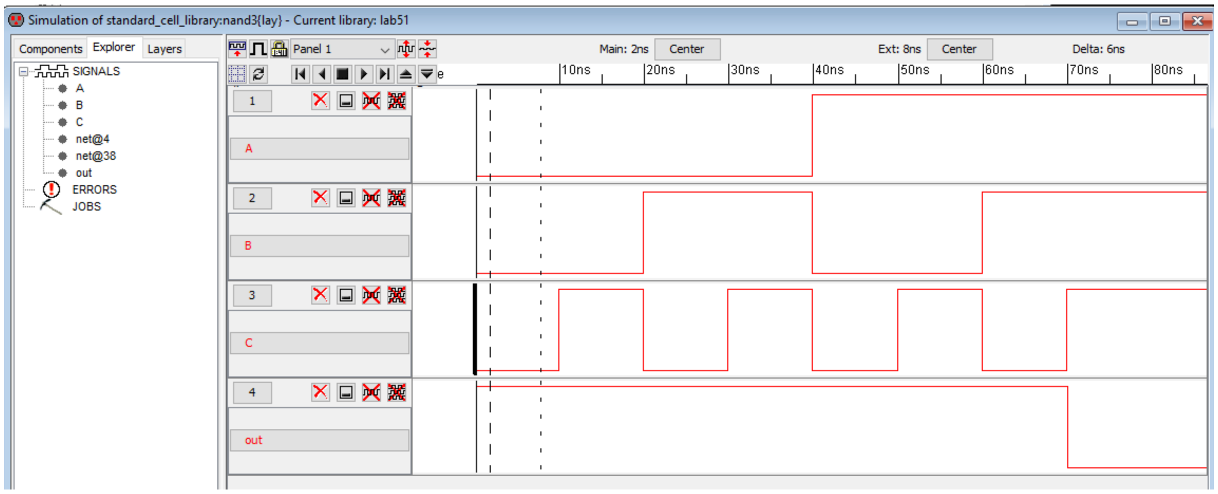
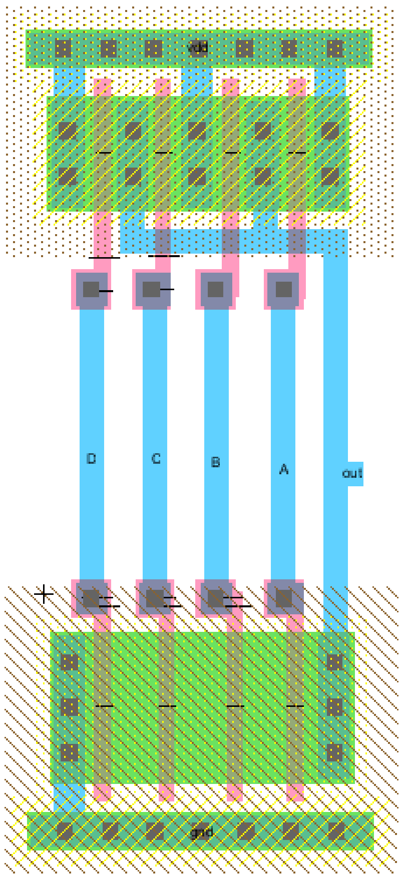
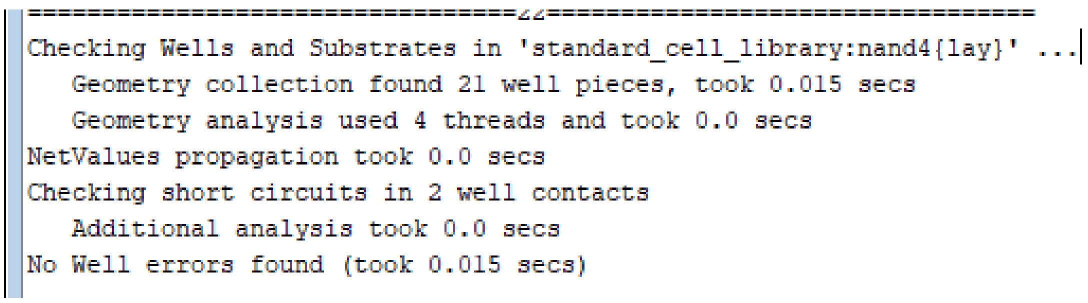


Figure 19 NAND3 Gate Layout Error Panel and Simulation Verification





```
=================================22=================================
Checking Wells and Substrates in 'standard_cell_library:nand4{lay}' ...
   Geometry collection found 21 well pieces, took 0.015 secs
   Geometry analysis used 4 threads and took 0.0 secs
NetValues propagation took 0.0 secs
Checking short circuits in 2 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.015 secs)
```

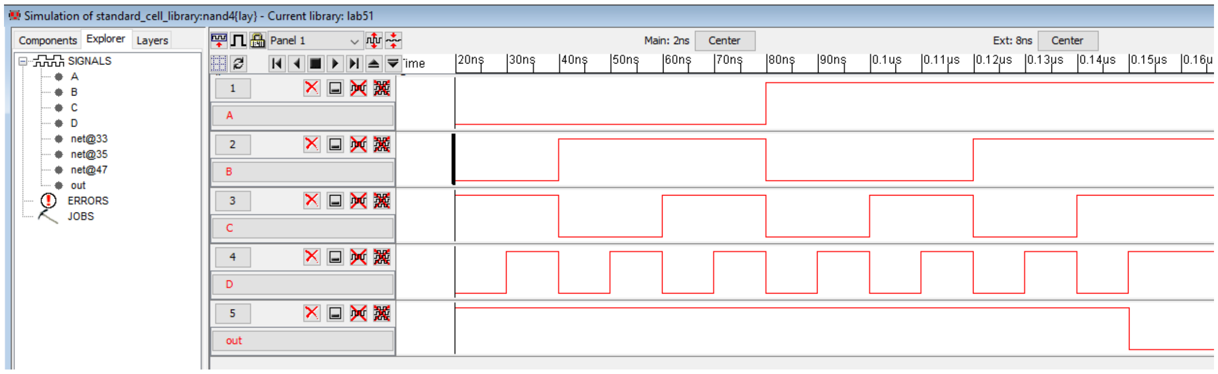
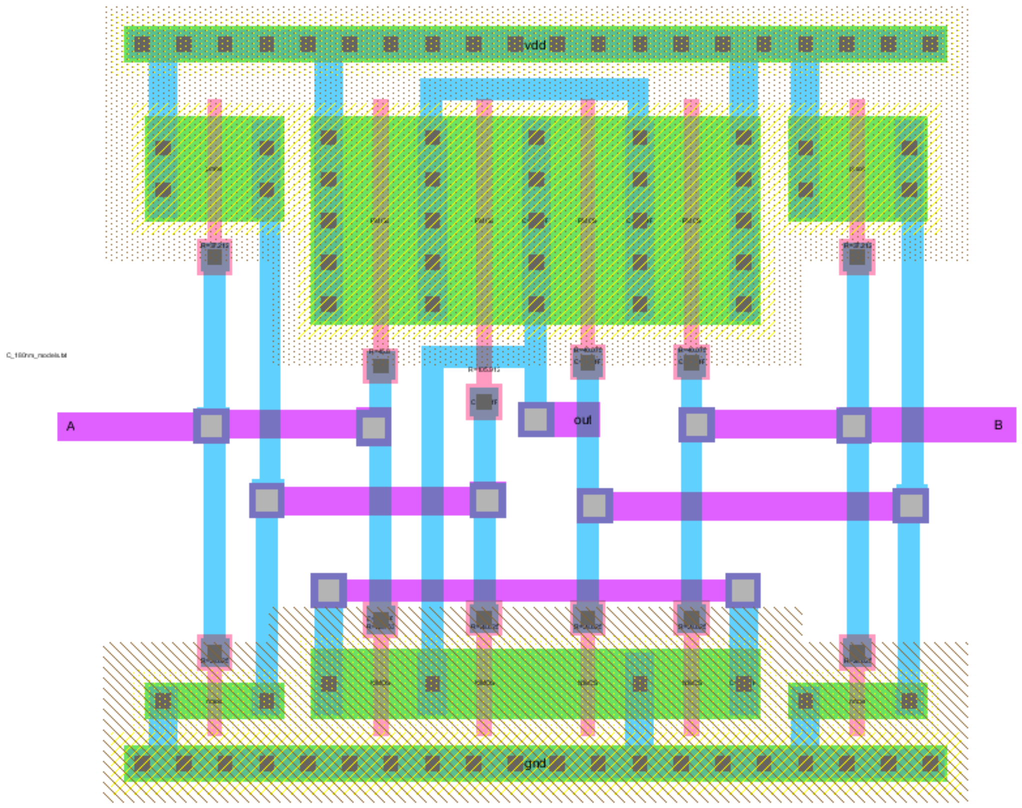
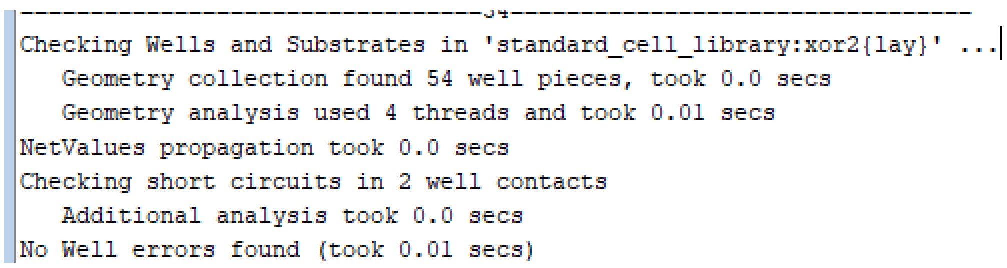


Figure 20 NAND4 Gate Layout Error Panel and Simulation Verification





```
Checking Wells and Substrates in 'standard_cell_library:xor2{lay}' ...
   Geometry collection found 54 well pieces, took 0.0 secs
   Geometry analysis used 4 threads and took 0.01 secs
NetValues propagation took 0.0 secs
Checking short circuits in 2 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.01 secs)
```

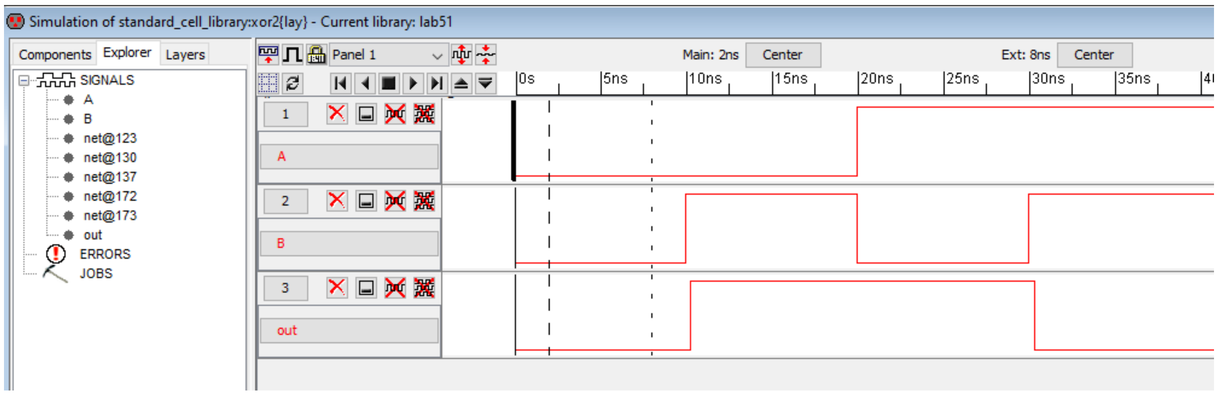
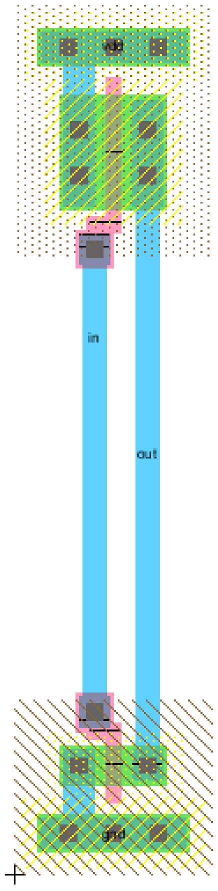
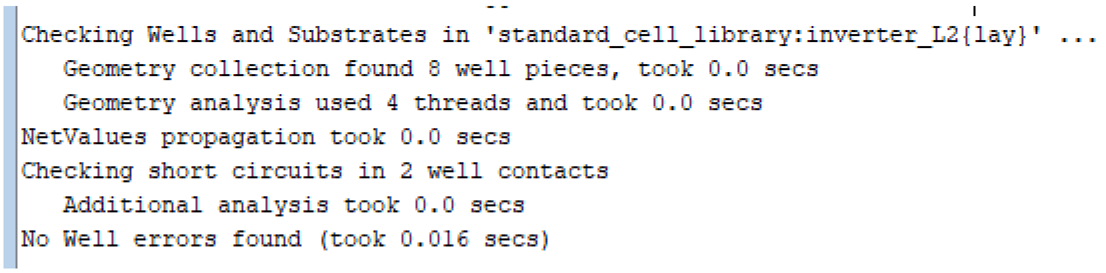


Figure 21 XOR2 Gate Layout Error Panel and Simulation Verification





```
Checking Wells and Substrates in 'standard_cell_library:inverter_L2{lay}' ...
   Geometry collection found 8 well pieces, took 0.0 secs
   Geometry analysis used 4 threads and took 0.0 secs
NetValues propagation took 0.0 secs
Checking short circuits in 2 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.016 secs)
```

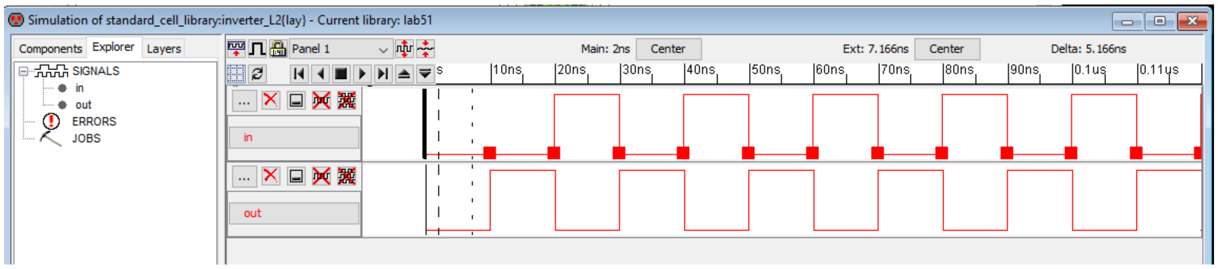
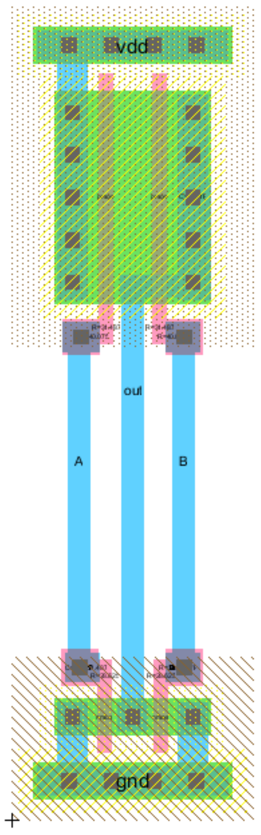
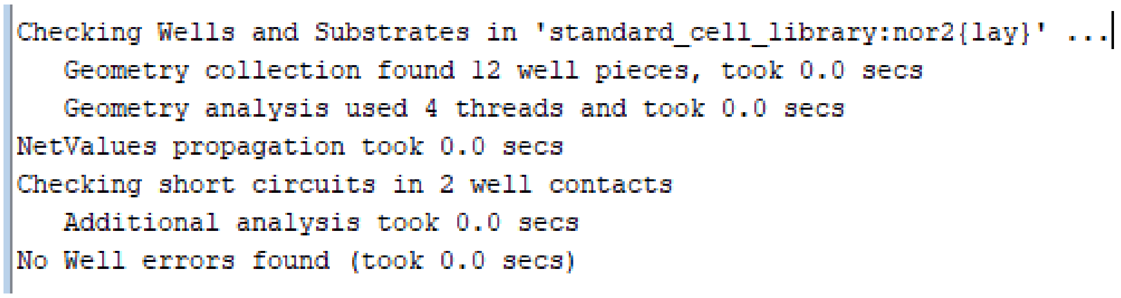

Simulation of standard_cell_library:inverter_L2{lay} - Current library: lab51

Figure 22 INVERTER Layout Error Panel and Simulation Verification





```
Checking Wells and Substrates in 'standard_cell_library:nor2{lay}' ...
    Geometry collection found 12 well pieces, took 0.0 secs
    Geometry analysis used 4 threads and took 0.0 secs
NetValues propagation took 0.0 secs
Checking short circuits in 2 well contacts
    Additional analysis took 0.0 secs
No Well errors found (took 0.0 secs)
```

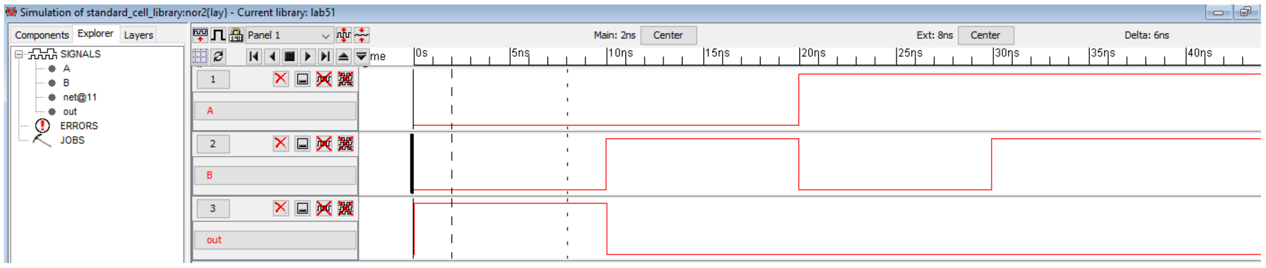
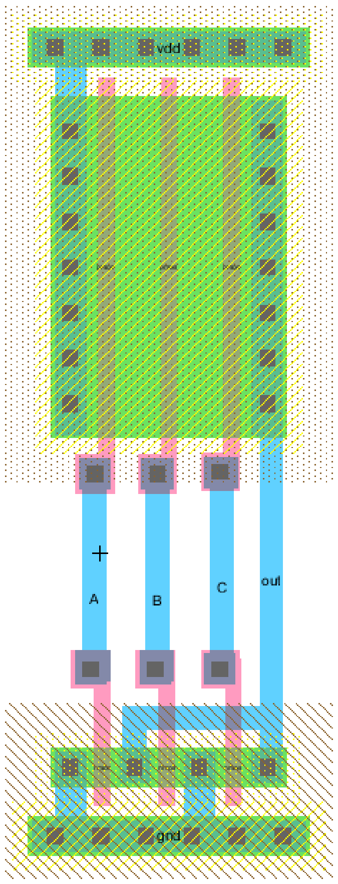
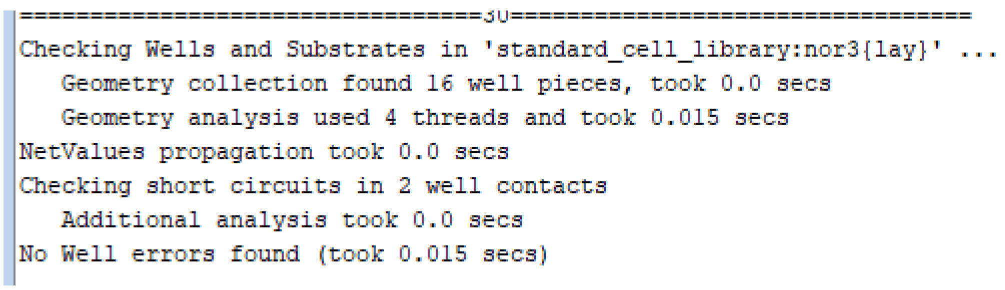


Figure 23 NOR2 Gate Layout Error Panel and Simulation Verification




```
===================================30==================================
Checking Wells and Substrates in 'standard_cell_library:nor3{lay}' ...
   Geometry collection found 16 well pieces, took 0.0 secs
   Geometry analysis used 4 threads and took 0.015 secs
NetValues propagation took 0.0 secs
Checking short circuits in 2 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.015 secs)
```

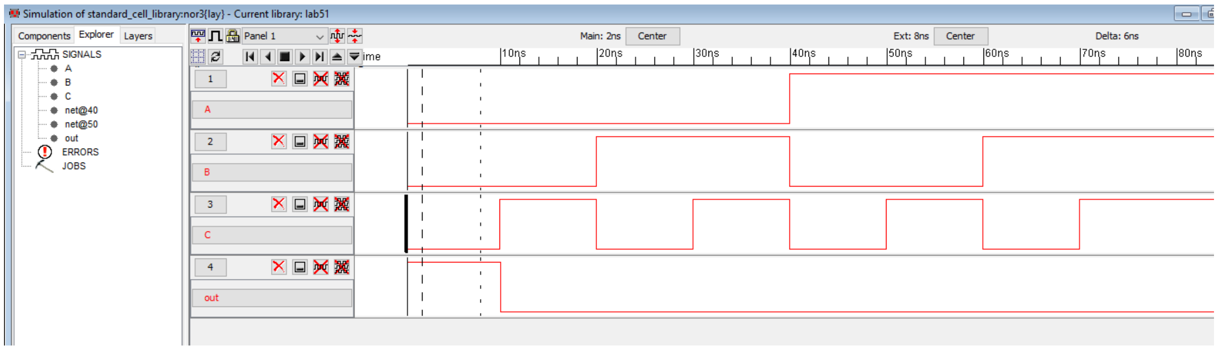


Figure 24 NOR3 Gate Layout Error Panel and Simulation Verification

APPENDIX C

5.1 CLA LAYOUTS

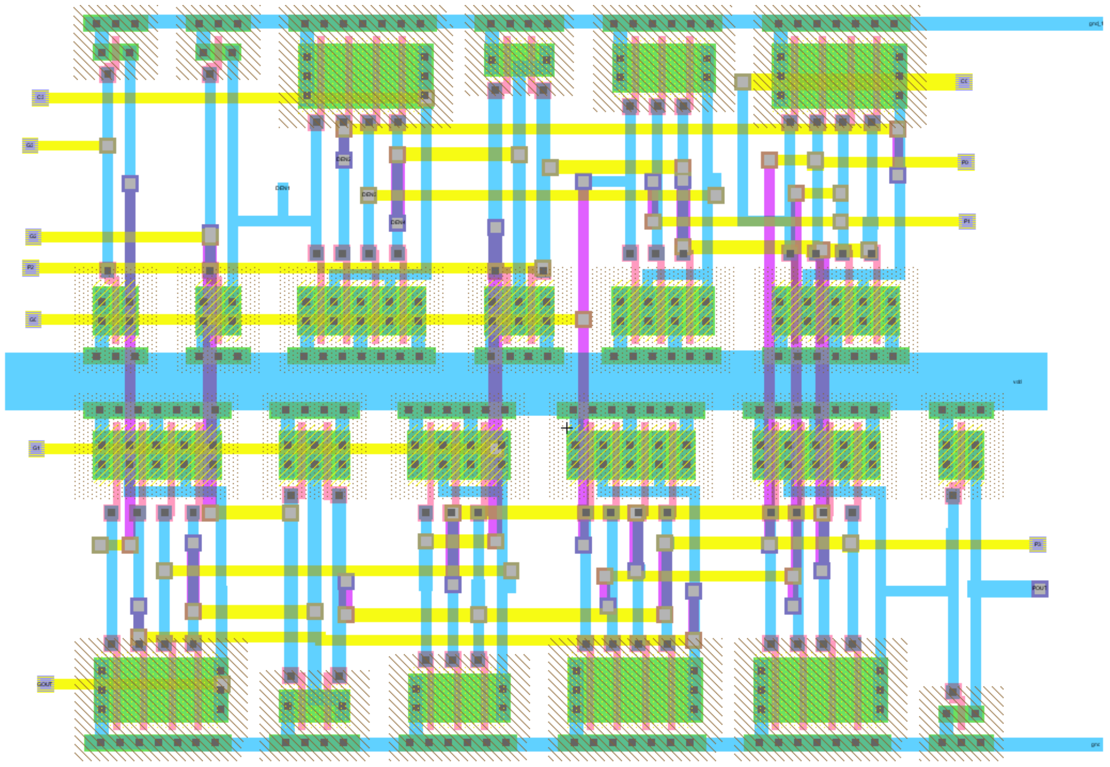
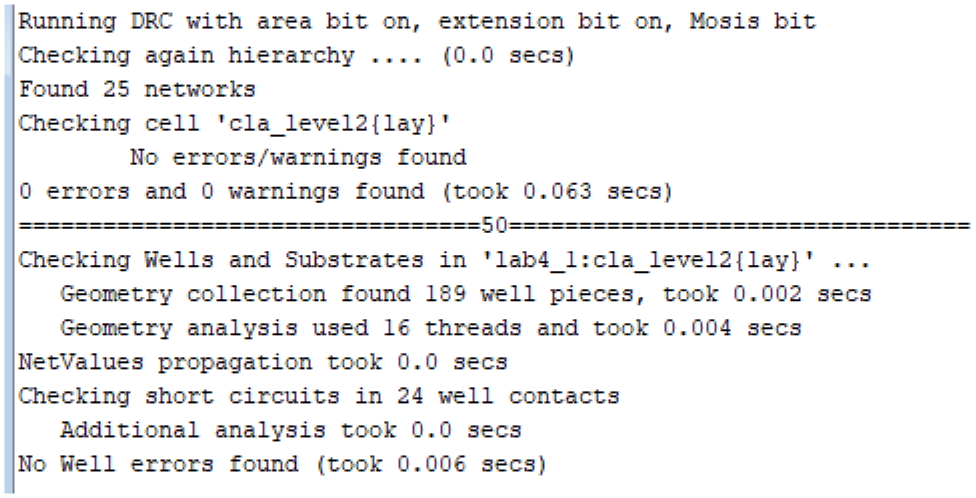


Figure 25 CLA Adder Level-2 Layout



```
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.0 secs)
Found 25 networks
Checking cell 'cla_level2{lay}'
        No errors/warnings found
0 errors and 0 warnings found (took 0.063 secs)
=================================50=================================
Checking Wells and Substrates in 'lab4_1:cla_level2{lay}' ...
   Geometry collection found 189 well pieces, took 0.002 secs
   Geometry analysis used 16 threads and took 0.004 secs
NetValues propagation took 0.0 secs
Checking short circuits in 24 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.006 secs)
```

Figure 26 CLA Adder Level-2 Error Panel

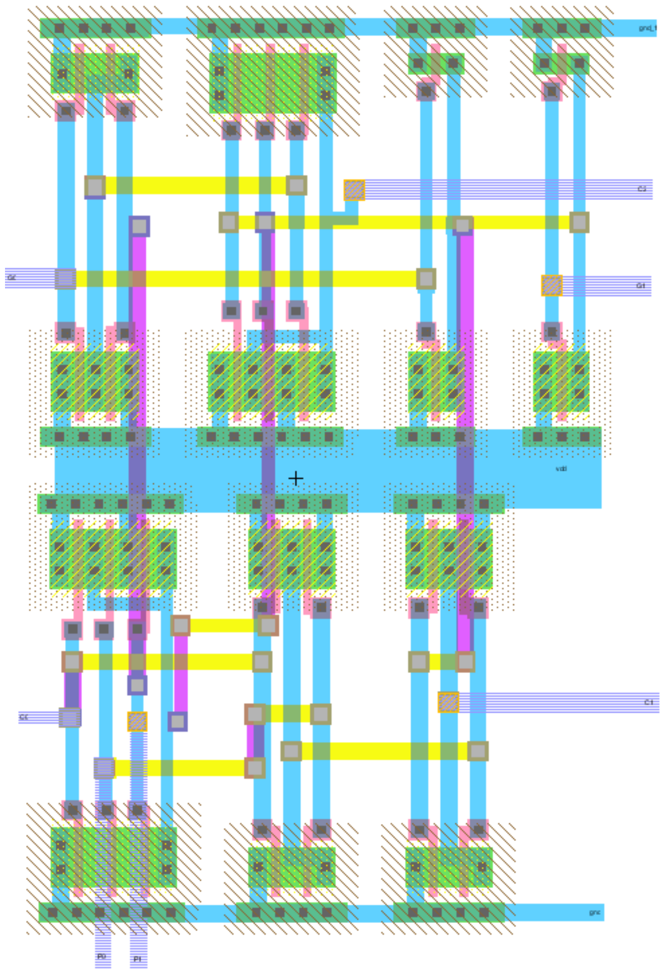
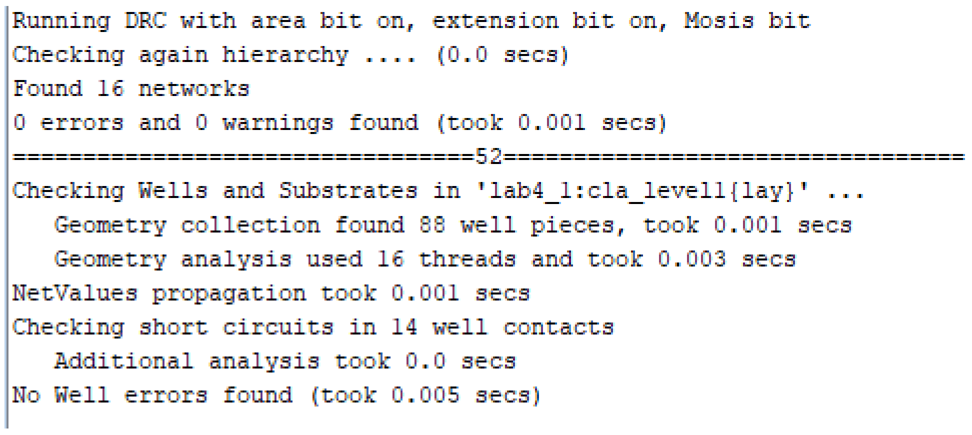


Figure 27 CLA Adder Level-1 Layout



```
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.0 secs)
Found 16 networks
0 errors and 0 warnings found (took 0.001 secs)
==================================52==================================
Checking Wells and Substrates in 'lab4_1:cla_level1{lay}' ...
   Geometry collection found 88 well pieces, took 0.001 secs
   Geometry analysis used 16 threads and took 0.003 secs
NetValues propagation took 0.001 secs
Checking short circuits in 14 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.005 secs)
```

Figure 28 CLA Adder Level-1 Error Panel

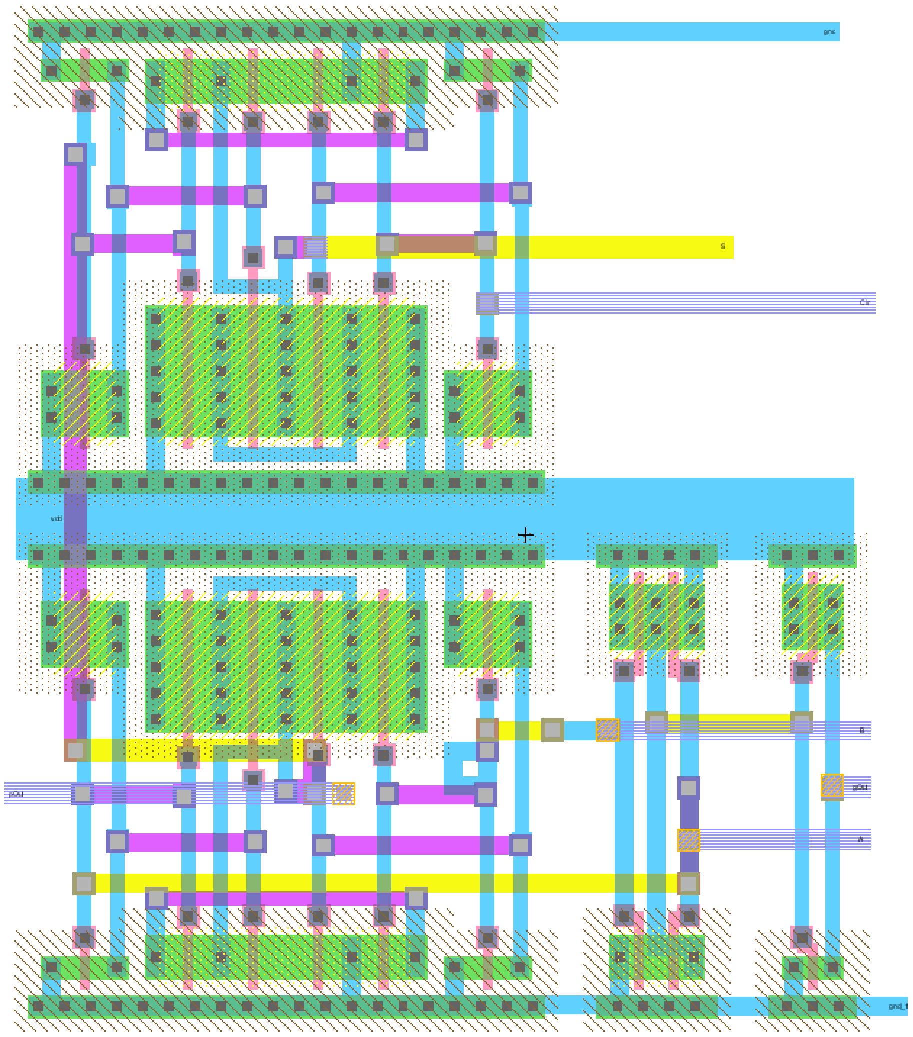
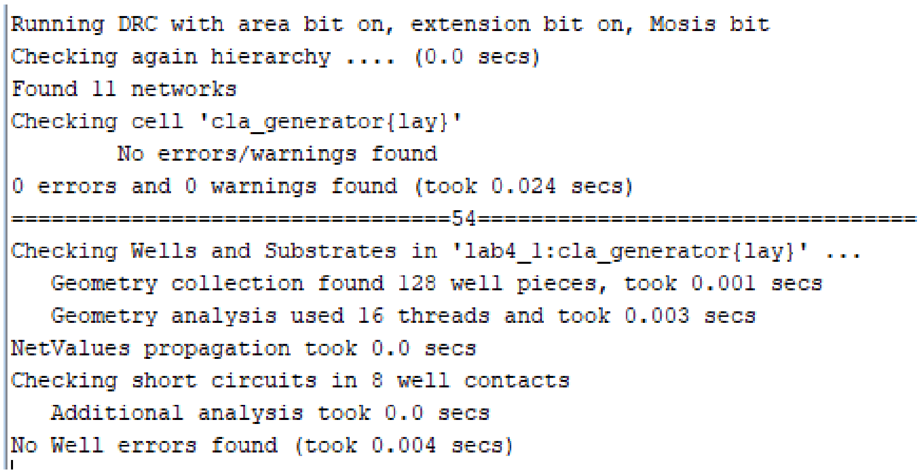


Figure 29 CLA Generator Layout



```
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.0 secs)
Found 11 networks
Checking cell 'cla_generator{lay}'
        No errors/warnings found
0 errors and 0 warnings found (took 0.024 secs)
=================================54=================================
Checking Wells and Substrates in 'lab4_1:cla_generator{lay}' ...
   Geometry collection found 128 well pieces, took 0.001 secs
   Geometry analysis used 16 threads and took 0.003 secs
NetValues propagation took 0.0 secs
Checking short circuits in 8 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.004 secs)
```

Figure 30 CLA Generator Error Panel

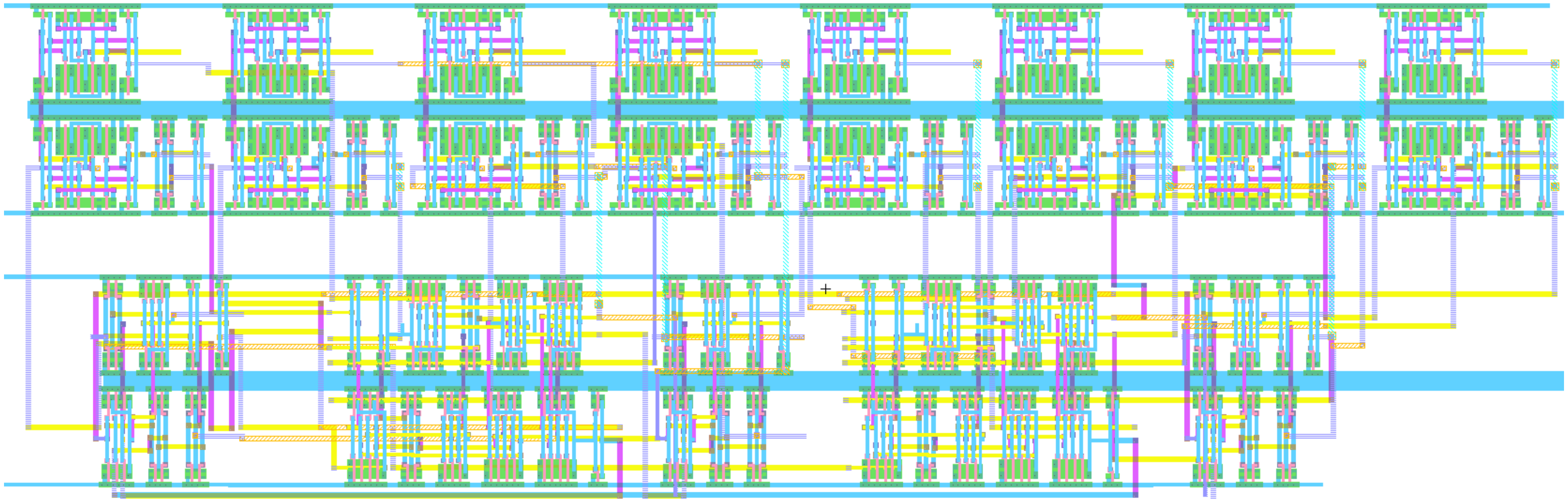
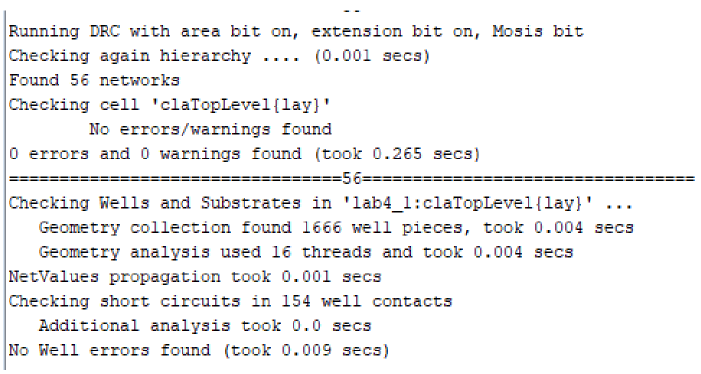


Figure 31 CLA Adder Design



```
                                --
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.001 secs)
Found 56 networks
Checking cell 'claTopLevel{lay}'
        No errors/warnings found
0 errors and 0 warnings found (took 0.265 secs)
=================================56=================================
Checking Wells and Substrates in 'lab4_1:claTopLevel{lay}' ...
   Geometry collection found 1666 well pieces, took 0.004 secs
   Geometry analysis used 16 threads and took 0.004 secs
NetValues propagation took 0.001 secs
Checking short circuits in 154 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.009 secs)
```

Figure 32 CLA Adder Error Panel

5.2 CLA WAVEFORMS

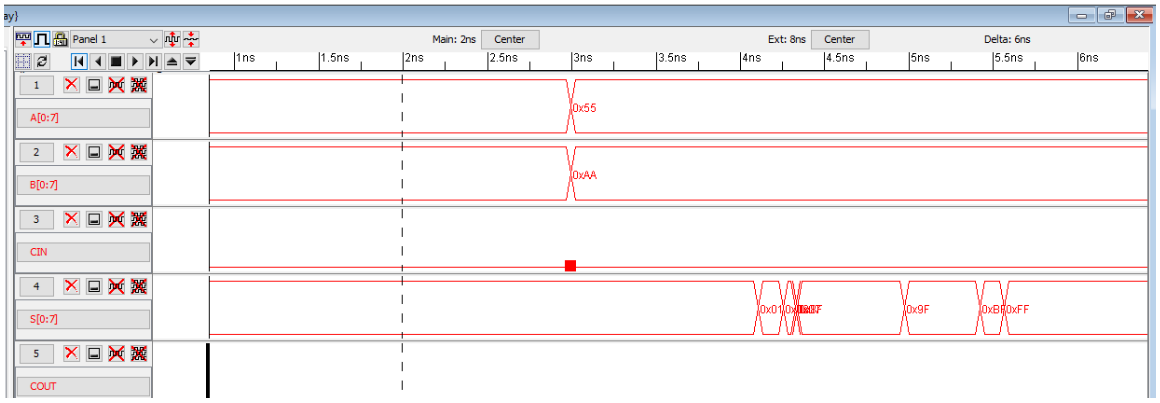


Figure 33 A:55 B:AA Cin:0 = S:FF COUT:0

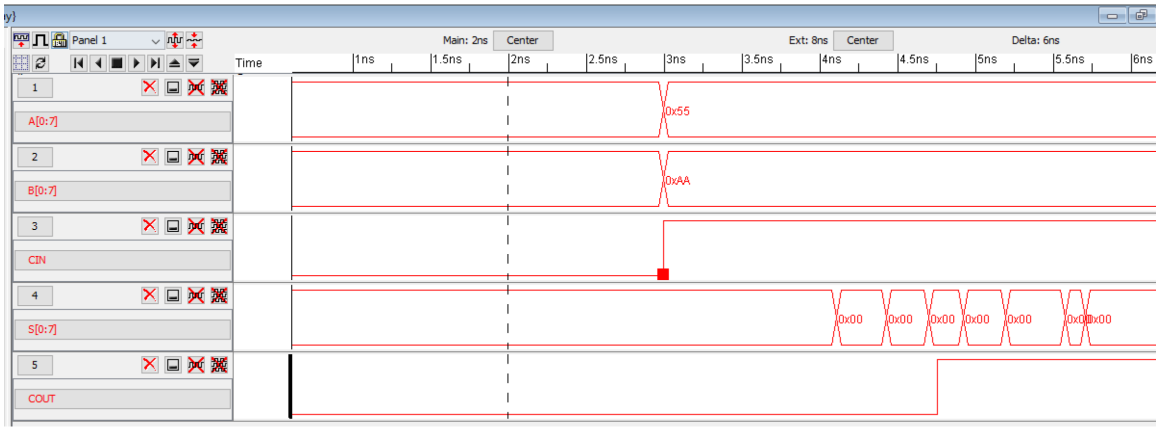


Figure 34 A:55 B:AA Cin:1 = S:00 COUT:1

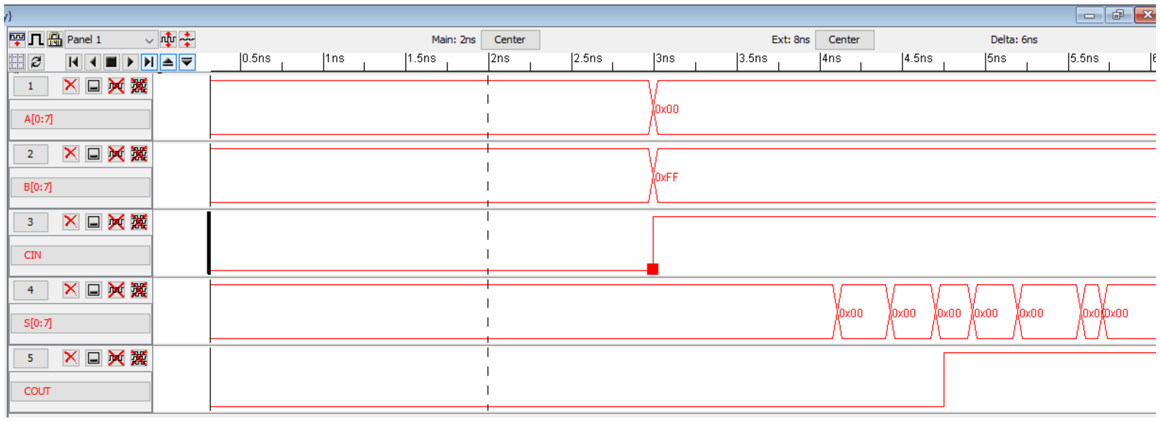


Figure 35 A:00 B:FF Cin:1 = S:00 COUT:1

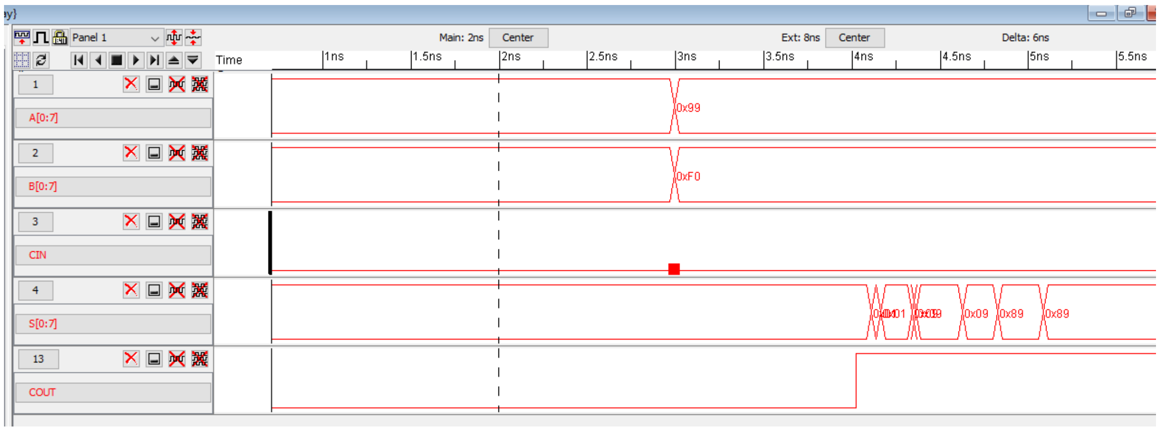


Figure 36 A:99 B:F0 Cin:0 = S:89 COUT:1

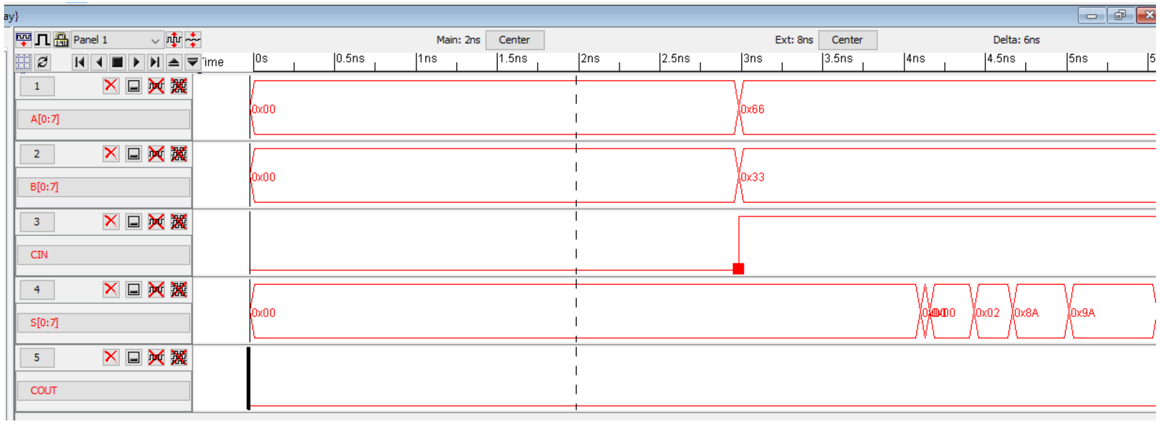


Figure 37 A:66 B:33 Cin:1 = S:9A COUT:0

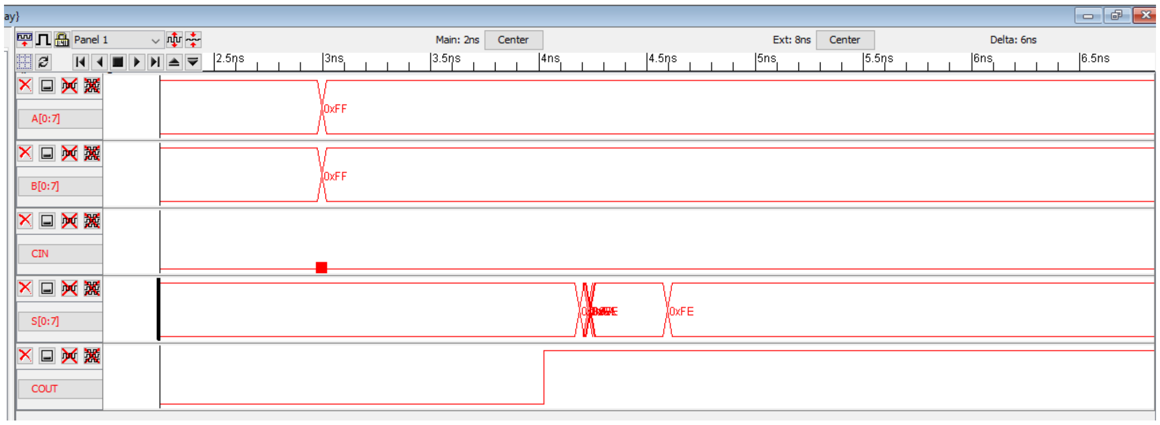


Figure 38 A:FF B:FF Cin:0 = S:FE COUT:1

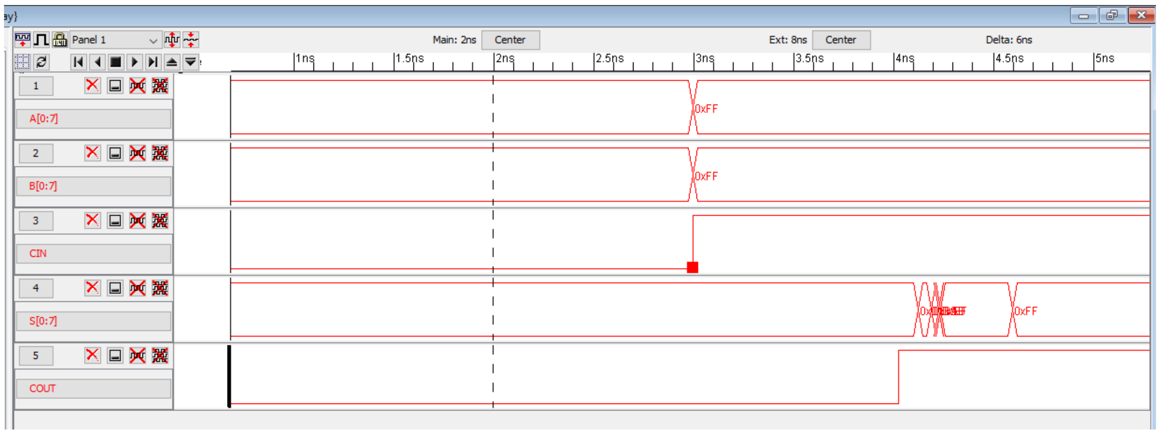


Figure 39 A:FF B:FF Cin:1 = S:FF COUT:1

5.3 INPUT REGISTER LAYOUTS

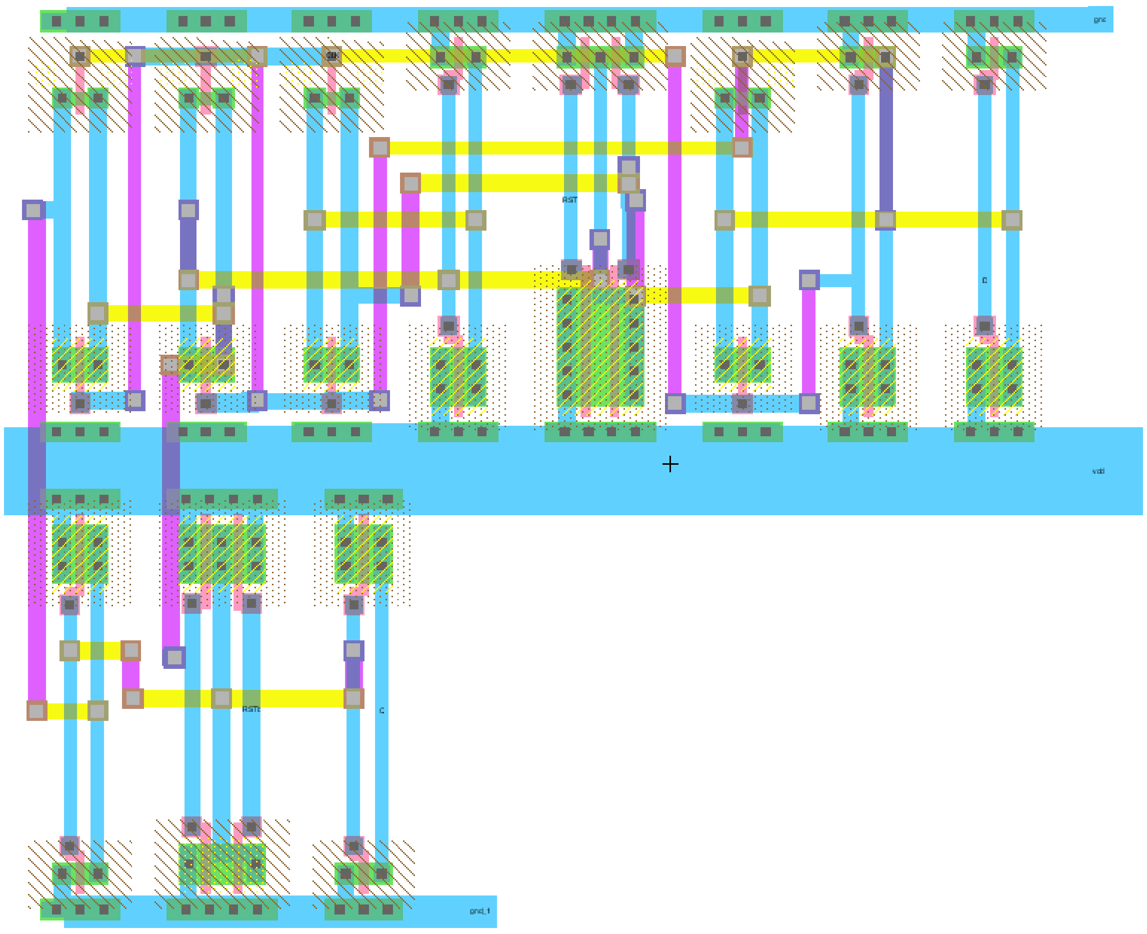
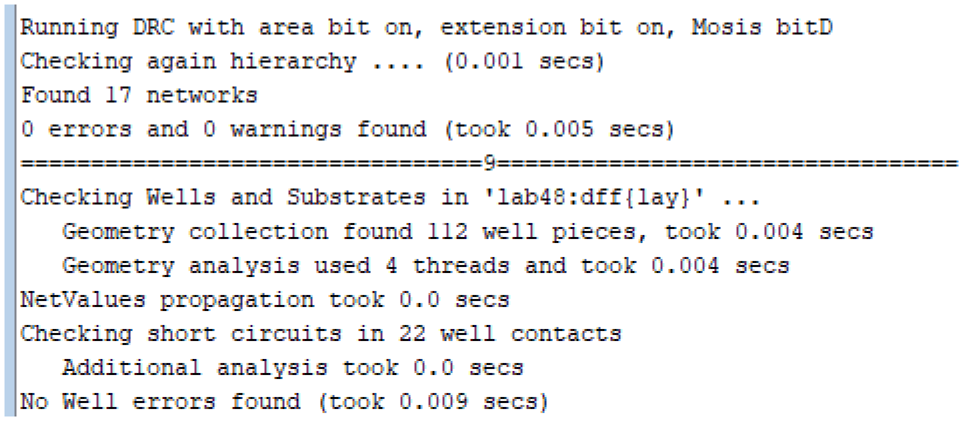


Figure 40 Flip Flop Layout



```
Running DRC with area bit on, extension bit on, Mosis bitD
Checking again hierarchy .... (0.001 secs)
Found 17 networks
0 errors and 0 warnings found (took 0.005 secs)
===================================9==================================
Checking Wells and Substrates in 'lab48:dff{lay}' ...
   Geometry collection found 112 well pieces, took 0.004 secs
   Geometry analysis used 4 threads and took 0.004 secs
NetValues propagation took 0.0 secs
Checking short circuits in 22 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.009 secs)
```

Figure 41 FlipFlop Error Panel

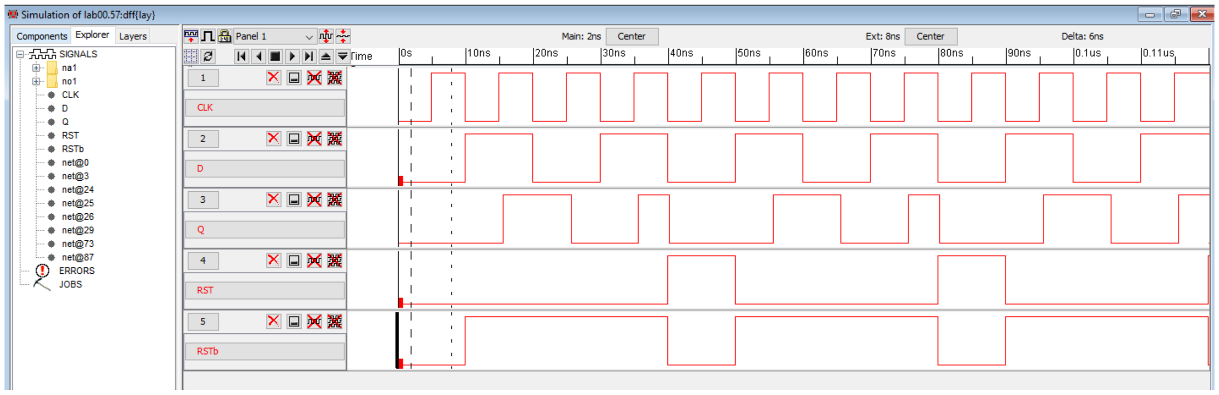
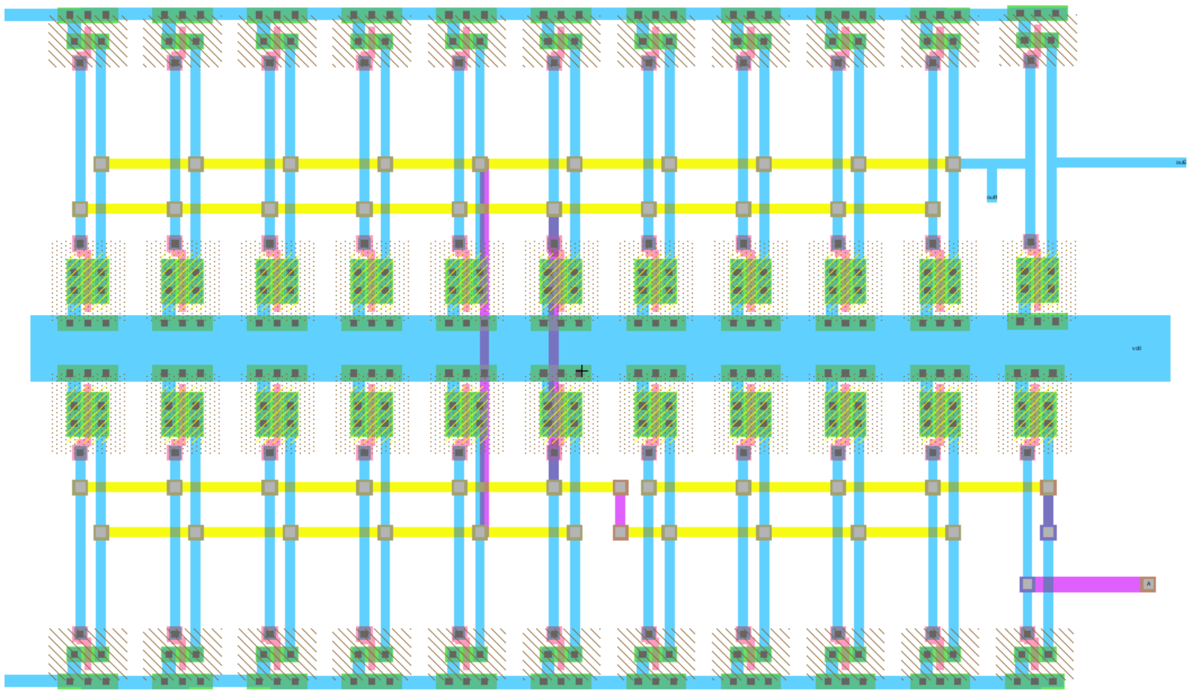
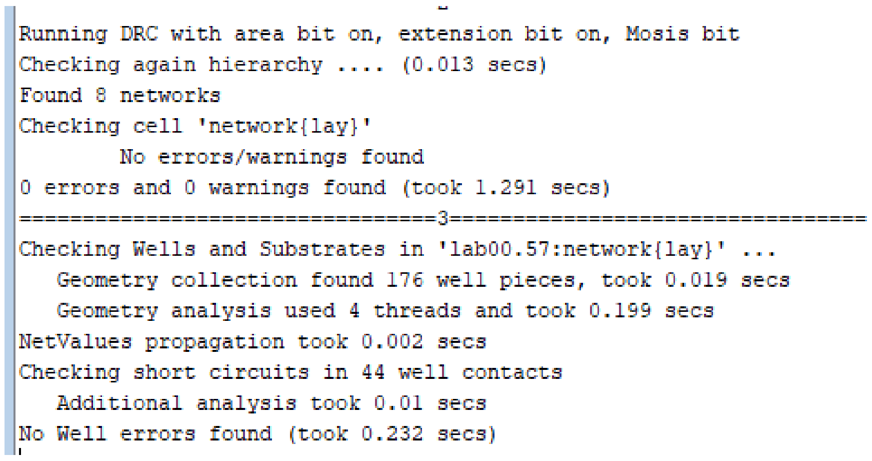


Figure 42 FlipFlop Simulation





```
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.013 secs)
Found 8 networks
Checking cell 'network{lay}'
        No errors/warnings found
0 errors and 0 warnings found (took 1.291 secs)
=================================3================================
Checking Wells and Substrates in 'lab00.57:network{lay}' ...
   Geometry collection found 176 well pieces, took 0.019 secs
   Geometry analysis used 4 threads and took 0.199 secs
NetValues propagation took 0.002 secs
Checking short circuits in 44 well contacts
   Additional analysis took 0.01 secs
No Well errors found (took 0.232 secs)
```

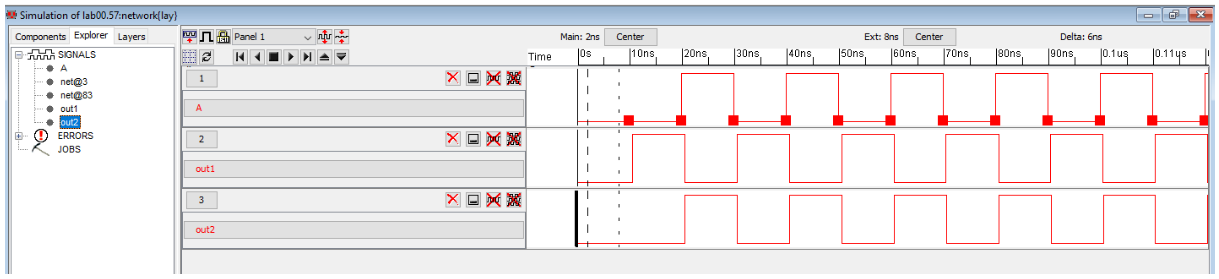
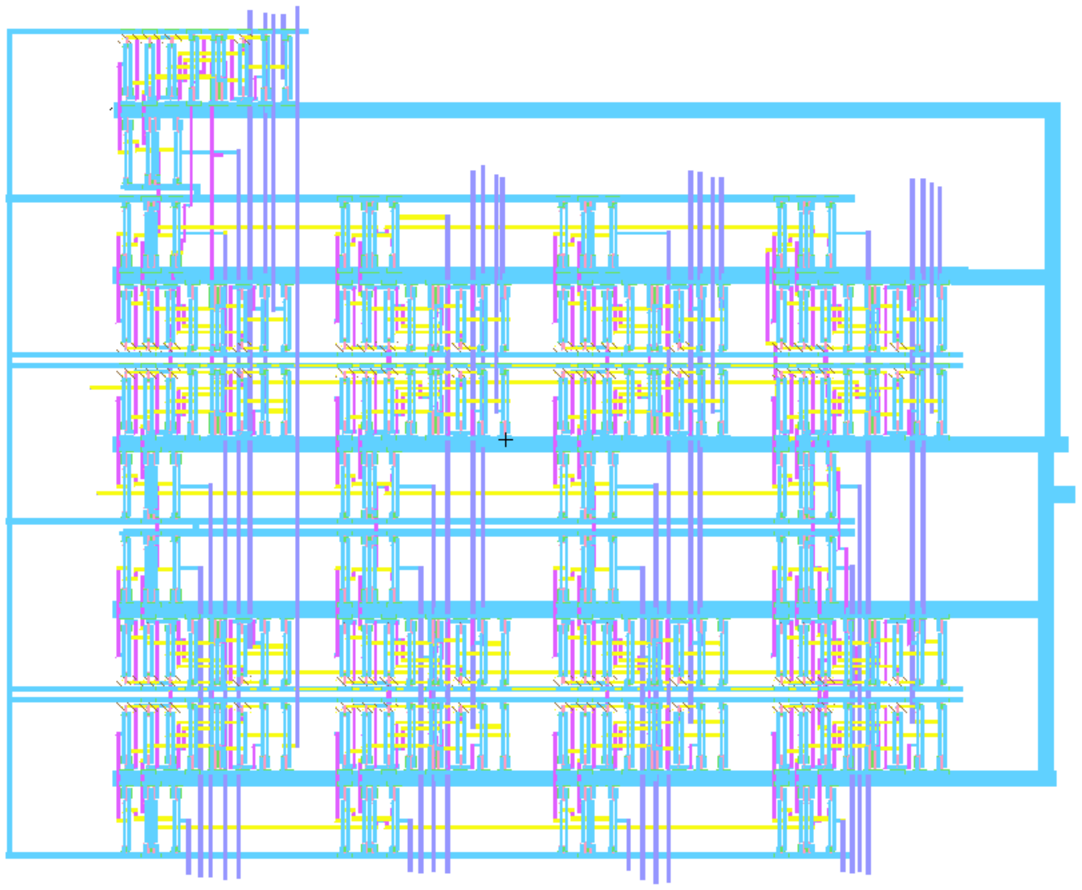
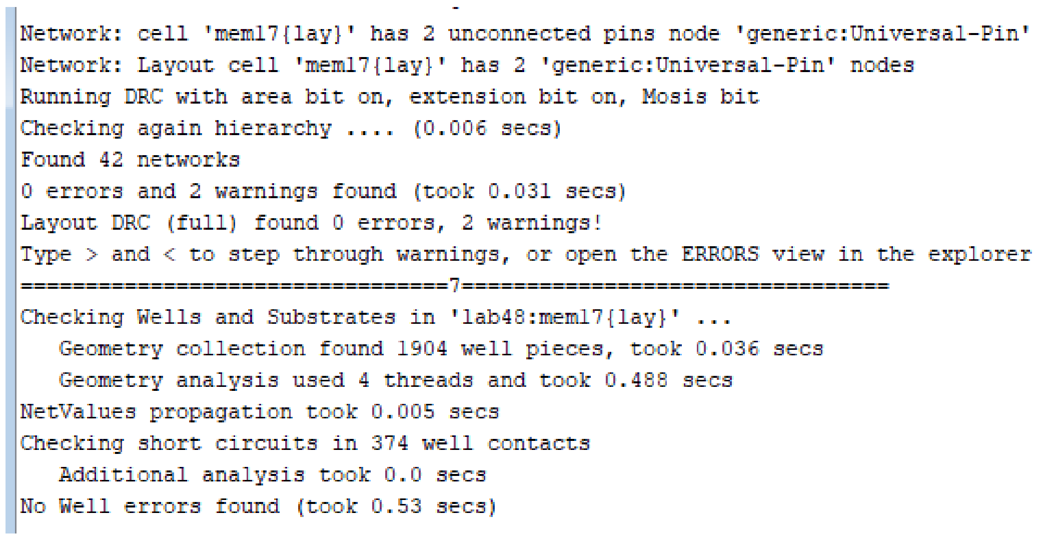


Figure 43 CLK Chain for 17 bit Register: Layout Error Panel and Simulation





```
Network: cell 'mem17{lay}' has 2 unconnected pins node 'generic:Universal-Pin'
Network: Layout cell 'mem17{lay}' has 2 'generic:Universal-Pin' nodes
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.006 secs)
Found 42 networks
0 errors and 2 warnings found (took 0.031 secs)
Layout DRC (full) found 0 errors, 2 warnings!
Type > and < to step through warnings, or open the ERRORS view in the explorer
=================================7=================================
Checking Wells and Substrates in 'lab48:mem17{lay}' ...
   Geometry collection found 1904 well pieces, took 0.036 secs
   Geometry analysis used 4 threads and took 0.488 secs
NetValues propagation took 0.005 secs
Checking short circuits in 374 well contacts
   Additional analysis took 0.0 secs
No Well errors found (took 0.53 secs)
```

Figure 44 17 bit Register Layout and Error Panel

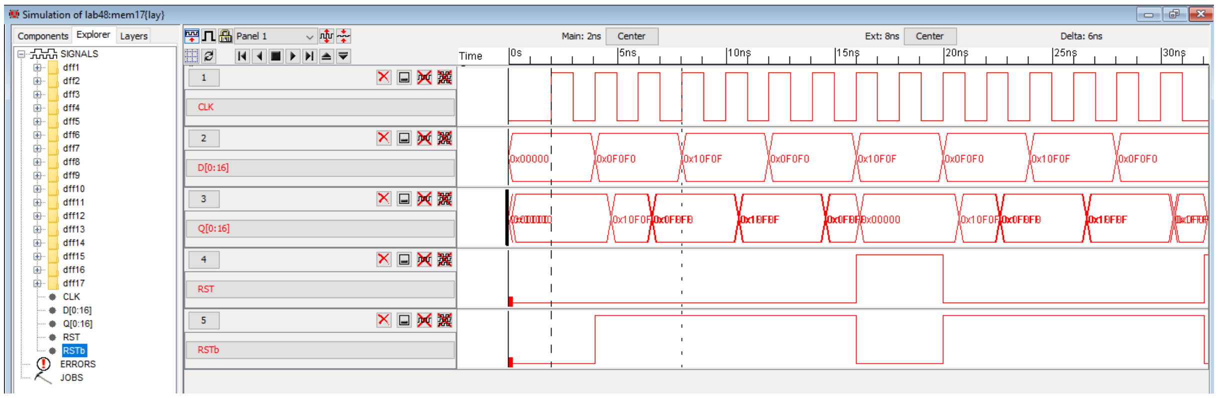
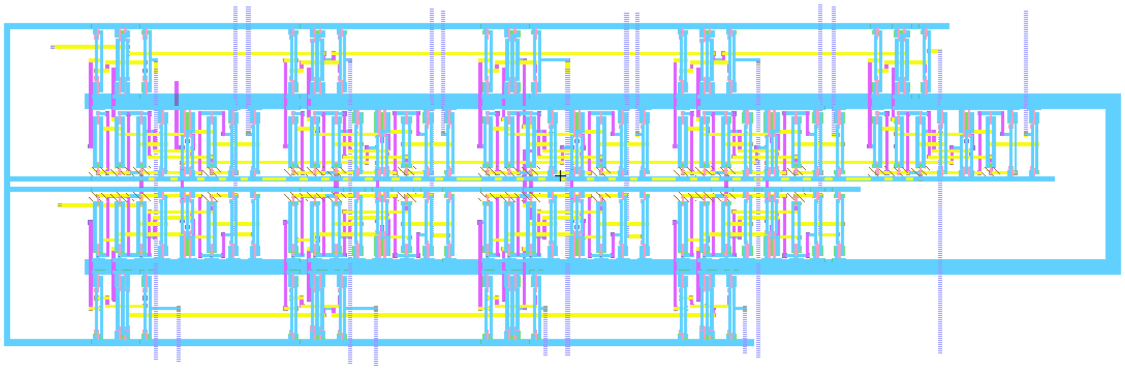
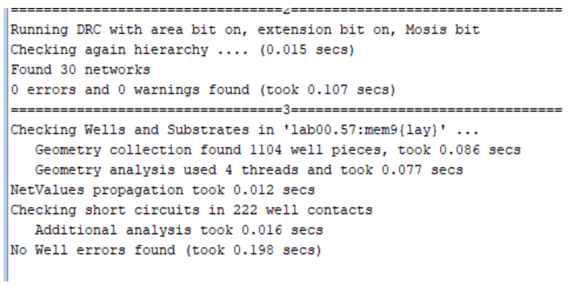


Figure 45 17 bit Register Simulation




```
=================================2=================================
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.015 secs)
Found 30 networks
0 errors and 0 warnings found (took 0.107 secs)
=================================3=================================
Checking Wells and Substrates in 'lab00.57:mem9{lay}' ...
   Geometry collection found 1104 well pieces, took 0.086 secs
   Geometry analysis used 4 threads and took 0.077 secs
NetValues propagation took 0.012 secs
Checking short circuits in 222 well contacts
   Additional analysis took 0.016 secs
No Well errors found (took 0.198 secs)
```

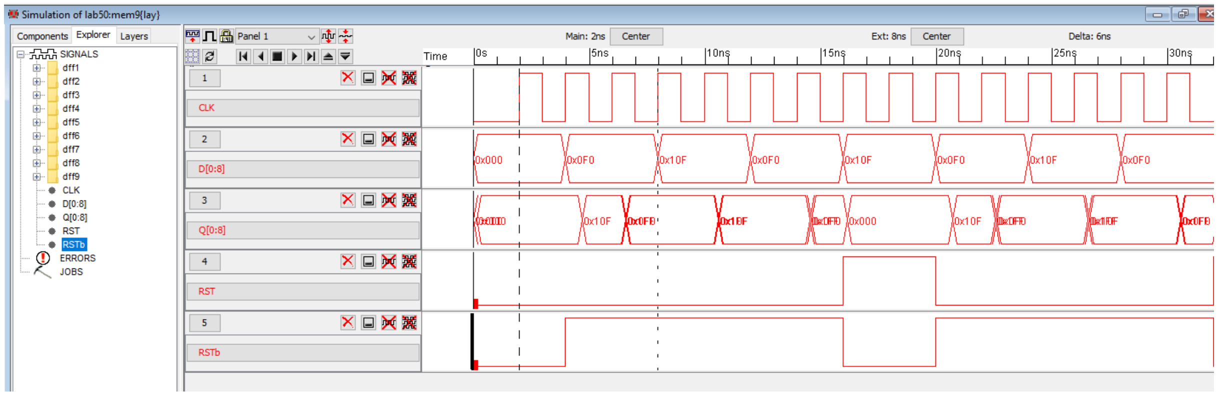
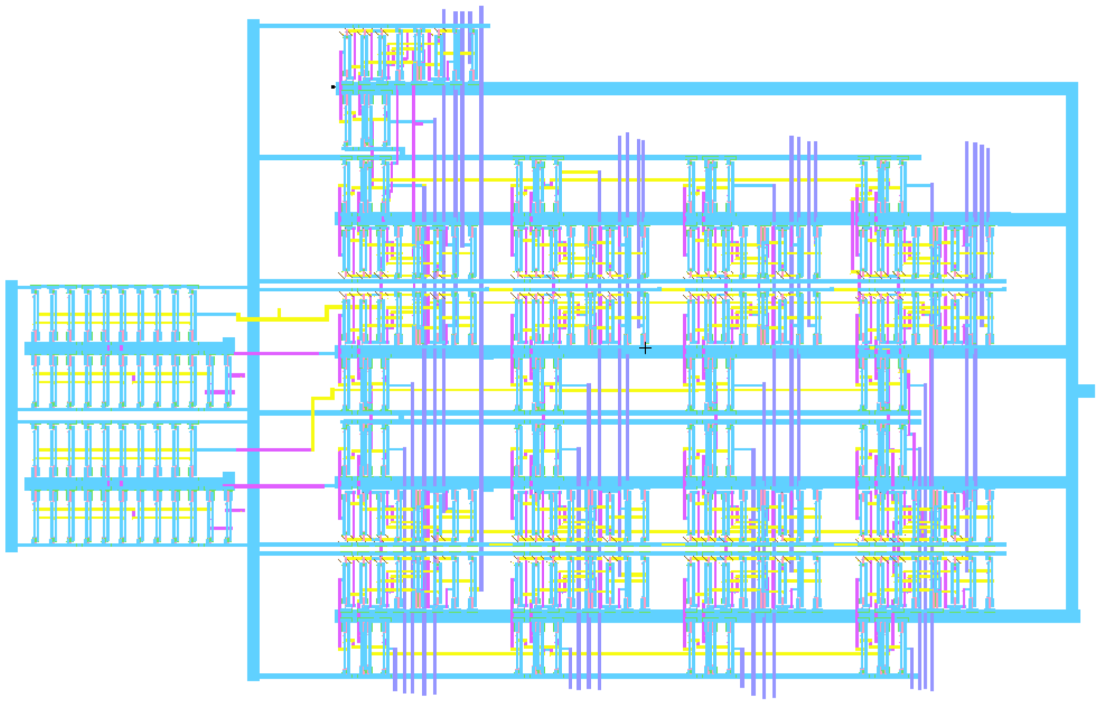
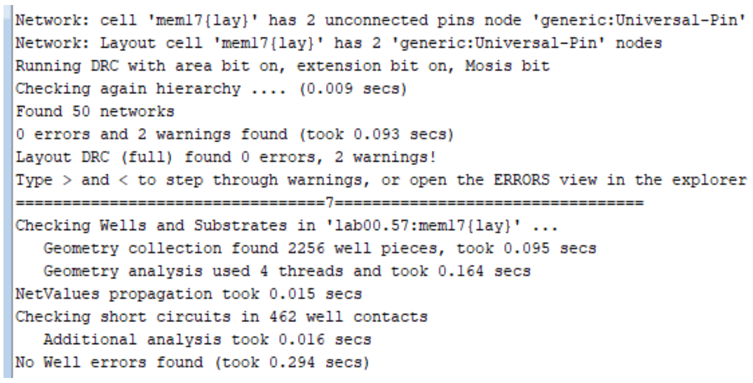


Figure 46 9 bit Register : Layout Error Panel and Simulation





```
Network: cell 'mem17{lay}' has 2 unconnected pins node 'generic:Universal-Pin'
Network: Layout cell 'mem17{lay}' has 2 'generic:Universal-Pin' nodes
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.009 secs)
Found 50 networks
0 errors and 2 warnings found (took 0.093 secs)
Layout DRC (full) found 0 errors, 2 warnings!
Type > and < to step through warnings, or open the ERRORS view in the explorer
================================7=================================
Checking Wells and Substrates in 'lab00.57:mem17{lay}' ...
   Geometry collection found 2256 well pieces, took 0.095 secs
   Geometry analysis used 4 threads and took 0.164 secs
NetValues propagation took 0.015 secs
Checking short circuits in 462 well contacts
   Additional analysis took 0.016 secs
No Well errors found (took 0.294 secs)
```

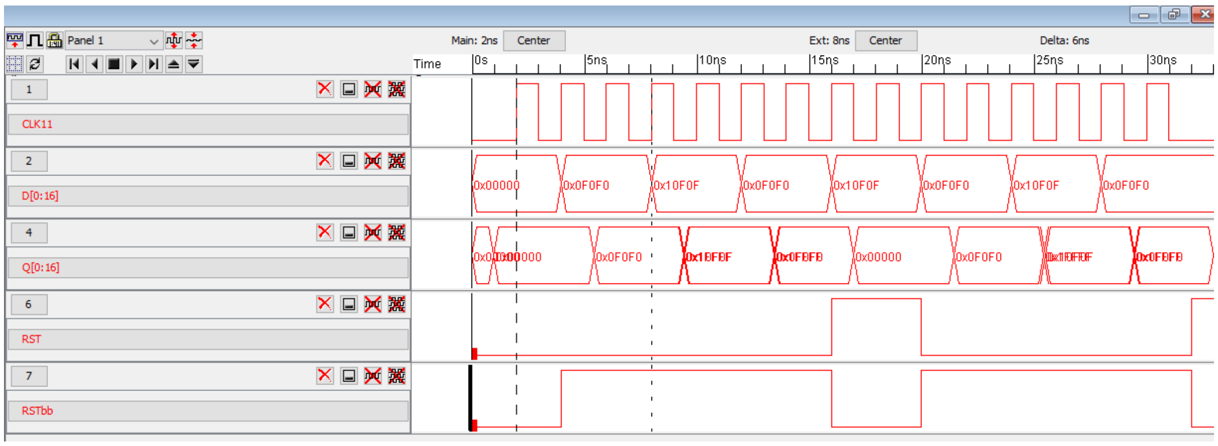
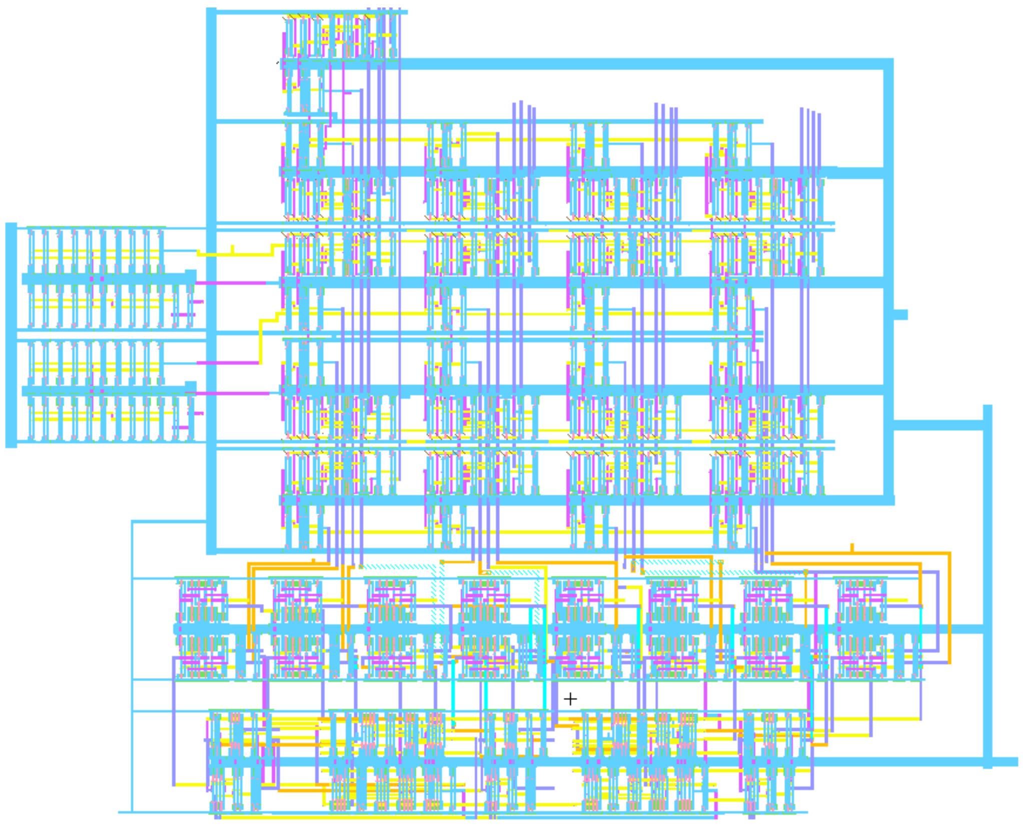
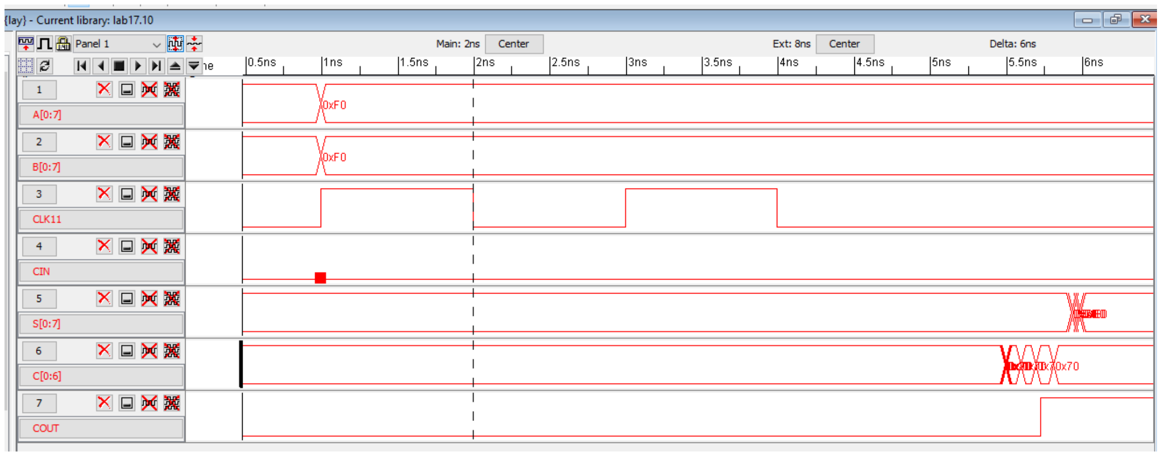


Figure 47 17 bit Register with Inverter Chains : Layout Error Panel and Simulation





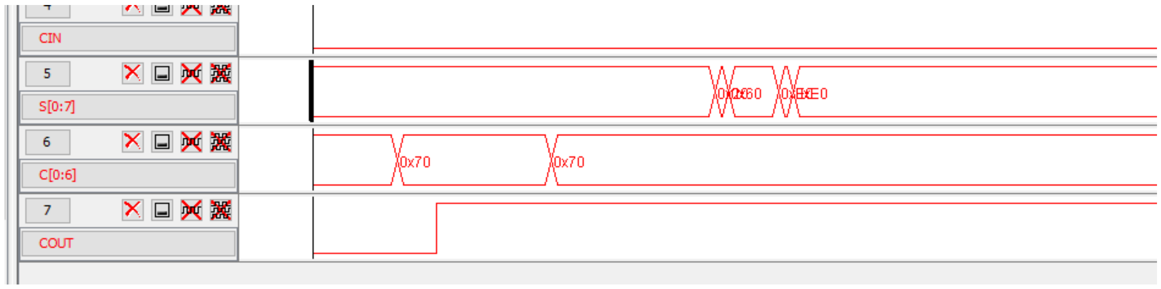
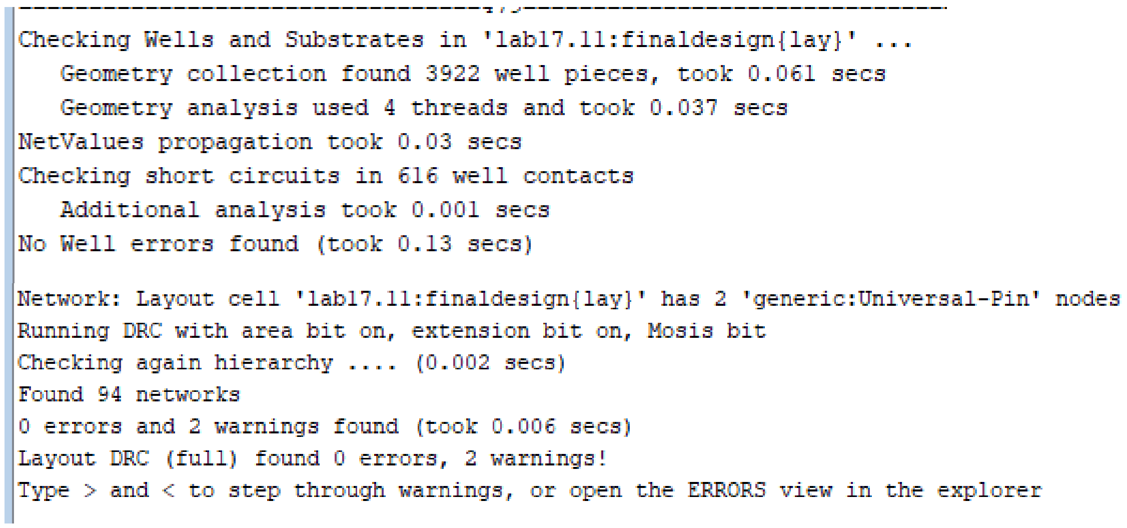


Figure 48 Full Circuit CLA Design: Layout and Simulation Verification



```
Checking Wells and Substrates in 'lab17.11:finaldesign{lay}' ...
   Geometry collection found 3922 well pieces, took 0.061 secs
   Geometry analysis used 4 threads and took 0.037 secs
NetValues propagation took 0.03 secs
Checking short circuits in 616 well contacts
   Additional analysis took 0.001 secs
No Well errors found (took 0.13 secs)

Network: Layout cell 'lab17.11:finaldesign{lay}' has 2 'generic:Universal-Pin' nodes
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.002 secs)
Found 94 networks
0 errors and 2 warnings found (took 0.006 secs)
Layout DRC (full) found 0 errors, 2 warnings!
Type > and < to step through warnings, or open the ERRORS view in the explorer
```

Figure 49 Full Circuit CLA Design Error Panel