

**Group Member: Qiaoya Cui & Guoyan Kai**  
**Section: C**

## **Project 3: Burglar Alarm System**

### ***Introduction:***

This final project is named Burglar alarm system. The objective of this project is to design a protection system for a house. In this protection circuit, there are six different sensors, once these sensors are satisfied the trigger condition. The burglar alarm will be sound. Therefore, the only way to turn off the alarm is enter the correct code by pushing buttons within 15 second. The code has five digits long and each digit has four bits long. Otherwise, if the user does not enter the wrong code in 15 second, the alarm will automatically call the police, which mean the output police will be triggered.

### ***Procedure:***

For the process of this project, there are major two steps which are based on the Verilog coding and schematic/ layout design respectively.

#### **1. Procedure One: Verilog Part:**

The Verilog code contains all programing step of this burglar alarm system, which could be able to control this burglar alarm by entering different inputs to this system. The design presented here concludes all the modules needed to implement to realize the working of this burglar alarm. There are four distinct modules and the names and functions list in below table.

Module Name	Function of the Module
BurglarAlarm2	Main module, combine all the modules together
decoder_button	A decoder to decode the button into binary number
compare	Compare the code user entered with the actual code of system
timer15s	Checking user entering time, whether it is over 15 seconds

##### **a. decoder\_button module:**

In this module, the main purpose is changed the 9 bit button input into binary number to make computer which number the user actually entered and read it out. The input of this module is Codein which presents the number user entered and the out will contain the entire numbers user entered about 20 bits.

```

module decoder_button (out, Codein);
    input [9:0] Codein;
    output [19:0] out;

    reg [19:0] out;
    reg [3:0] decode_out; // decoded number;
    reg [3:0] register[4:0]; //store decoded numbers each time user entered;
    reg [2:0] counter = 0; // count user just enter five digits number;

always @ (Codein) begin
    case (Codein)
        10'b00000000001: decode_out = 4'd0; //button 0
        10'b00000000010: decode_out = 4'd1; //button 1
        10'b00000000100: decode_out = 4'd2; //button 2
        10'b00000001000: decode_out = 4'd3; //button 3
        10'b00000010000: decode_out = 4'd4; //button 4
        10'b00000100000: decode_out = 4'd5; //button 5
        10'b00001000000: decode_out = 4'd6; //button 6
        10'b00010000000: decode_out = 4'd7; //button 7
        10'b00100000000: decode_out = 4'd8; //button 8
        10'b01000000000: decode_out = 4'd9; //button 9
    endcase
    if (counter < 5) begin // enter times
        register[counter] = decode_out;
        counter = counter + 1;
    end
    out = {register[4], register[3], register[2], register[1], register[0]};
    // put total five numbers into a single register which has 20 bits. |
end
endmodule

```

## b. compare module:

This module is used to compare the correctness between the real code and code user entered. The real code has been set up in this module.

```

module compare (PASSout, PASSin);
    input [19:0] PASSin; // code user entered
    output PASSout; // correctness of the code user entered;
    reg out; // register to store the PASSout value;

    // The real five digits code number.
    parameter PASS0 = 4'b0001;
    parameter PASS1 = 4'b0010;
    parameter PASS2 = 4'b0011;
    parameter PASS3 = 4'b0100;
    parameter PASS4 = 4'b0101;

    always @ (PASSin) begin
        // check the two group of code one by one;
        if (PASSin[3:0] != PASS0 || PASSin[7:4] != PASS1 || PASSin[11:8] != PASS2
            || PASSin[15:12] != PASS3 || PASSin[19:16] != PASS4)
        begin
            out = 0;
        end

        else begin
            out = 1;
        end
    end

    assign PASSout = out;
endmodule

```

## c. timer15s module:

This module is to make sure the user entered the correct code and the entering code time should be within 15 seconds. If not, the output of this module will be equal 0 which will be assigned into the main function and the POLICE output will be triggered, if here, the output is 1, in main function, both SOUND and POLICE will be turned into 0.

```

module timer15s(clk,sound, TIMERinput, TIMERoutput);

input clk;
input TIMERinput; //(Compare module result)-correctness of code user entered.
input sound; // SOUND input, SOUND =1 when trigger=1;
output TIMERoutput; // check result of the time limit.

reg out; // store the TIMERoutput result.
reg [5:0] count = 0; // counter the time untill 15 seconds.

always @ (posedge clk) begin

    if (sound == 1) begin
        count <= count + 1'd1; // check the time at each posedge of clk cycle;
        if (count != 6'd15) begin
            if (TIMERinput==1) // compare result
                out <= 1; // enter correct and within 15 seconds
            else if (TIMERinput==0 || count >= 6'd15) // entered wrong code or run out of 15s
                out <= 0; //
        end
    end
    assign TIMERoutput = out;
endmodule

```

## d. Main module - BurglarAlarm2

The main module includes the input and output of this burglar alarm system.

They are list in the below table:

Inputs	Outputs
WINDOW 1:3	SOUND
MOTION 1:3	POLICE
SET	
BUTTON 0:9	

In this main module, it has to connect all the modules together. Meanwhile, there are some states involved in this module. Each state will assign a different output results. The state result is list on the below stable:

State	The meaning of each state
00	SOUND = 0; POLICE = 0;
10	SOUND = 1 ; POLICE = 0;
11	SOUND = 1; POLICE =1;

The Verilog code of main module is shown below:

# EE 330 Final Project

December 9, 2011

```
`timescale 1s/10ms
////////////////////////////////////
module BurglarAlarm2 (SOUND,POLICE,
                     WINDOW1,WINDOW2,WINDOW3,MOTION1,MOTION2,MOTION3,
                     SET,
                     clk,
                     button);

input  WINDOW1,WINDOW2,WINDOW3,MOTION1,MOTION2,MOTION3;
input  SET;
input  [9:0] button;
input  clk;
output SOUND,POLICE;

reg SOUND;
reg POLICE;
reg [1:0] state;
wire trigger;
assign trigger = WINDOW1|WINDOW2|WINDOW3
                | (MOTION1&MOTION2|MOTION3&MOTION2|MOTION1&MOTION3);
reg controltrigger;
reg w1;
reg w2;
reg w3;
reg m1;
reg m2;
reg m3;

wire [19:0] checkout;
wire passout;
wire timeroutput;
// call another three module
decoder_button gate0(checkout,button);
compare_gate1(passout,checkout);
timer15s gate2(clk,trigger, passout, timeroutput);

always @ (posedge trigger) begin
    controltrigger <= trigger;
end

always @ (posedge clk) begin
if (!SET)// SET =0 means the system is off.
    begin
        //all the input should be 0 here.
        state <= 0;
        SOUND <= 0;
        POLICE <= 0;
        w1<=0;
        w2<=0;
        w3<=0;
        m1<=0;
        m2<=0;
        m3<=0;
    end
end
```

```

else if (SET) begin// SET =1 means the system is on.
case (state)// three different cases of state.
2'b00: begin SOUND<=0;
POLICE<=0;
if (controltrigger == 1) begin
state <=2'b10;
controltrigger = 0;

end
end

2'b10: begin SOUND<=1;
POLICE<=0;
if (passout == 1 && timeroutput==1) begin
// correct code and within 15s
state <= 2'b00;// turn to state 00
end

else if (passout == 0 || timeroutput==0 )begin
//wrong code entered or out of 15s
state <=2'b11;
end
end

2'b11: begin
SOUND<=1;
POLICE<=1;// POLICE is triggered.
end
endcase
end
end
endmodule

```

**e. Test Bench Code:**

This is the test bench of this Verilog. For testing it, there are three conditions which are correct code entering, incorrect code entering and run out of 15s. The time could be controlled by clk in test bench. The test bench shown below is for testing the time the user used to entering the code.

```

`timescale 100ms/10ms
module burglaralarm2_tb
wire SOUND;
wire POLICE;
reg SET;
reg WINDOW1;
reg WINDOW2;
reg WINDOW3;
reg MOTION2;
reg MOTION1;
reg MOTION3;
reg clk = 0;
reg [9:0] button;

```

```
BurglarAlarm2 BurglarAlarm2 (  
    .SOUND(SOUND),  
    .POLICE(POLICE),  
    .WINDOW1(WINDOW1),  
    .WINDOW2(WINDOW2),  
    .WINDOW3(WINDOW3),  
    .MOTION1(MOTION1),  
    .MOTION2(MOTION2),  
    .MOTION3(MOTION3),  
    .button(button),  
    .SET(SET),  
    .clk(clk)  
);  
  
always #5 clk = ~clk;  
initial  
begin  
    #0 SET = 1'b0;  
    #10 SET = 1'b1;  
    #5 WINDOW1=1'b0;WINDOW2=1'b0;WINDOW3=1'd0;MOTION1=1'b1;MOTION2=1'b1;MOTION3=1'b0;  
  
    #200 button = 10'b0000000001;  
    #5 button = 10'b00000000100;  
    #5 button = 10'b00000001000;  
    #5 button = 10'b00000010000;  
    #5 button = 10'b00000100000;  
    #5 button = 10'b00000000000;  
    #30;  
    $stop;  
  
end  
endmodule
```

***f. Waveform of test result:***

There are three different waveforms under three conditions:

- 1). triggered =1; code is correct;

# EE 330 Final Project

December 9, 2011

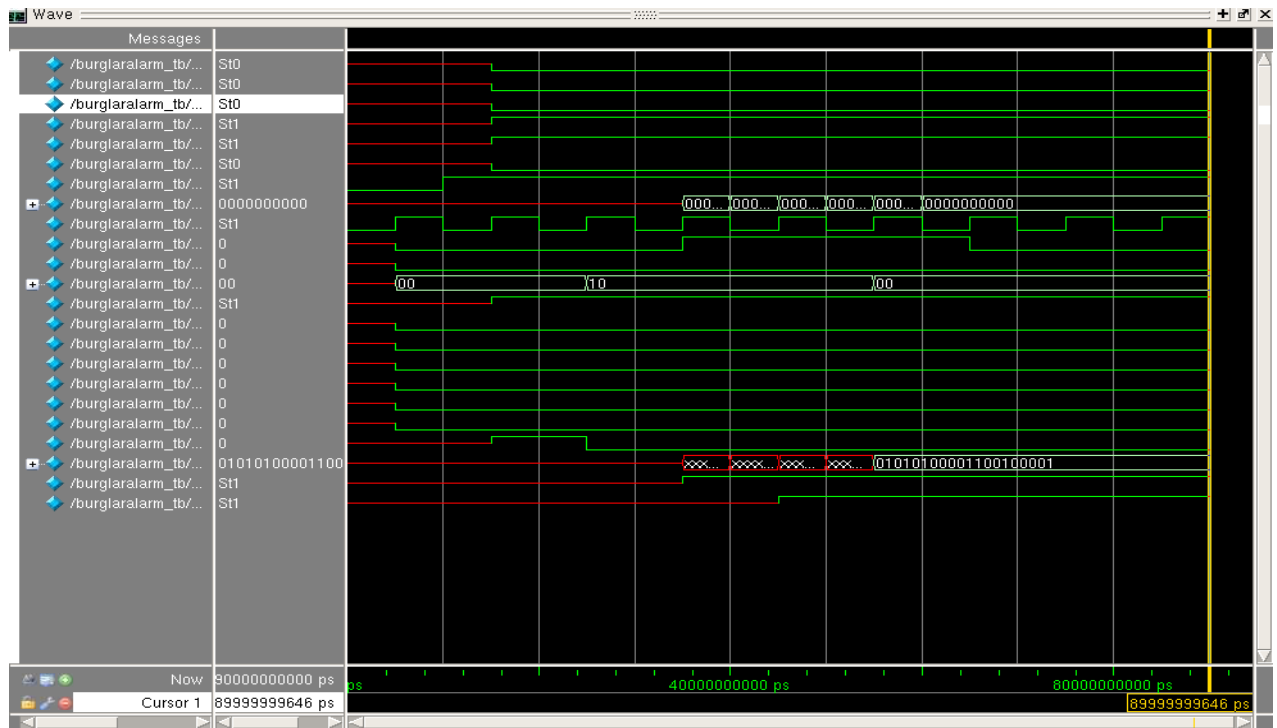


Figure 1: waveform for correct code

2). triggered =1; code is incorrect;

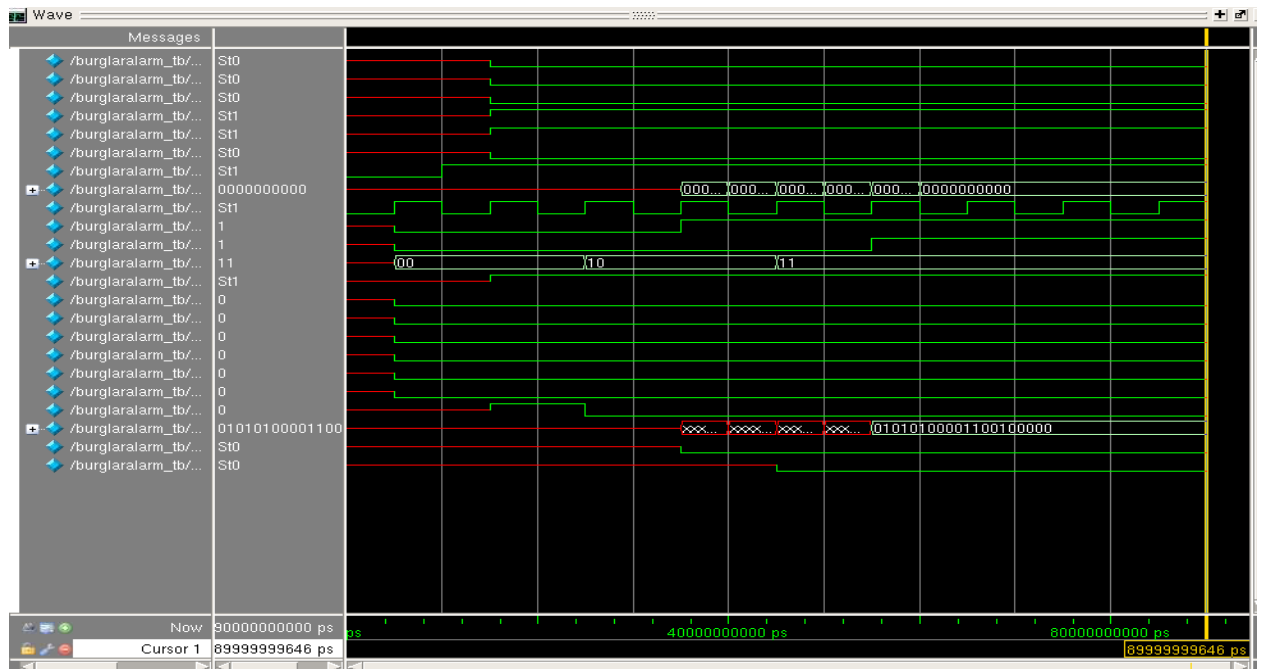


Figure 2: waveform for incorrect code

3). triggered =1; code does not entered in 15s

# EE 330 Final Project

December 9, 2011

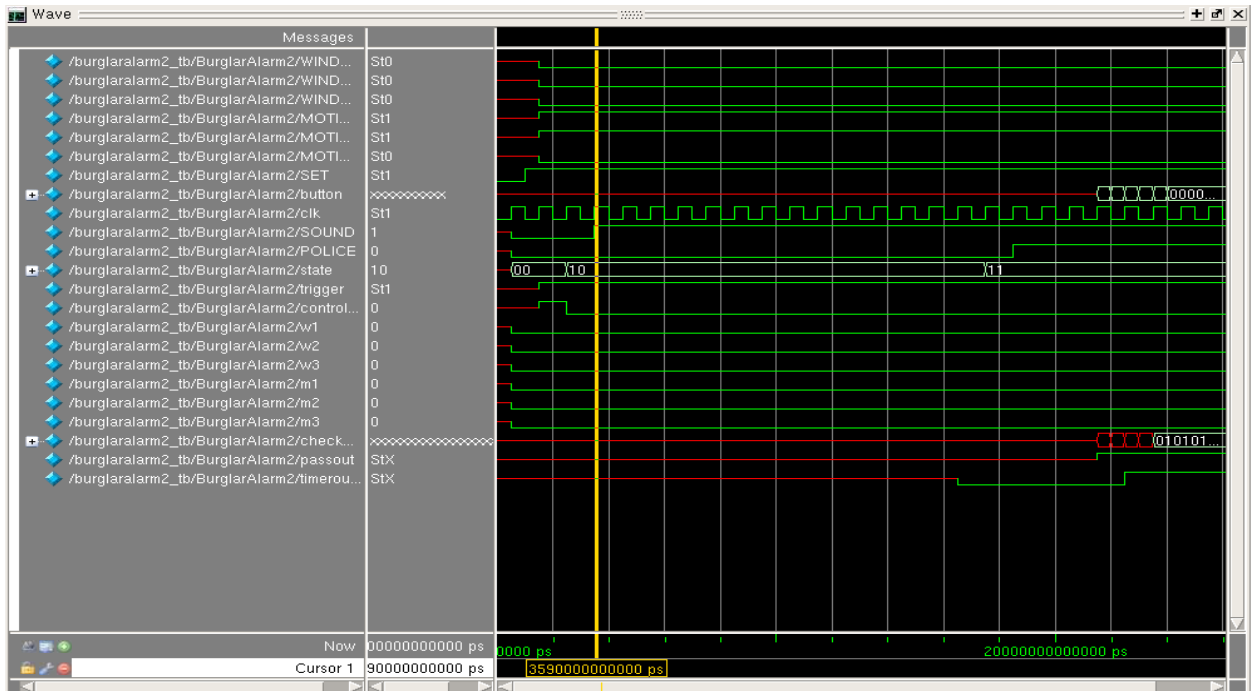


Figure 3: run out of 15s & the time sound is on

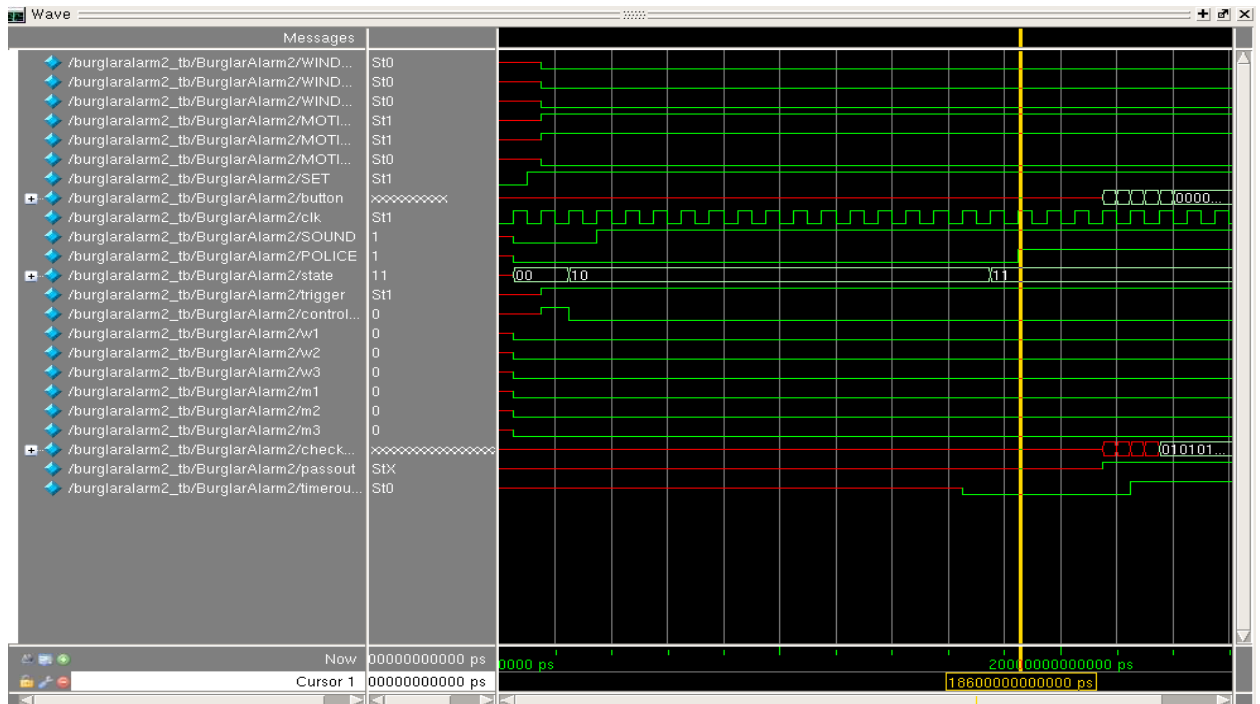


Figure 4: run out of 15s & police is on

From the observation, the time point of starting the sound is 3.59s and the time point of starting the police is 18.6s. Therefore, it can be proved that the duration time is 15s.



## 2. Schematic/ Layout Part:

### 1). Schematic:

After finished the Verilog file for this burglar alarm design, the next step is to draw out the schematic and construct the synthesis file. The procedure is similar to the previous lab, if follow the one by one step of that, it is not difficult to get the schematic circuit below. The only different step is library changing here.

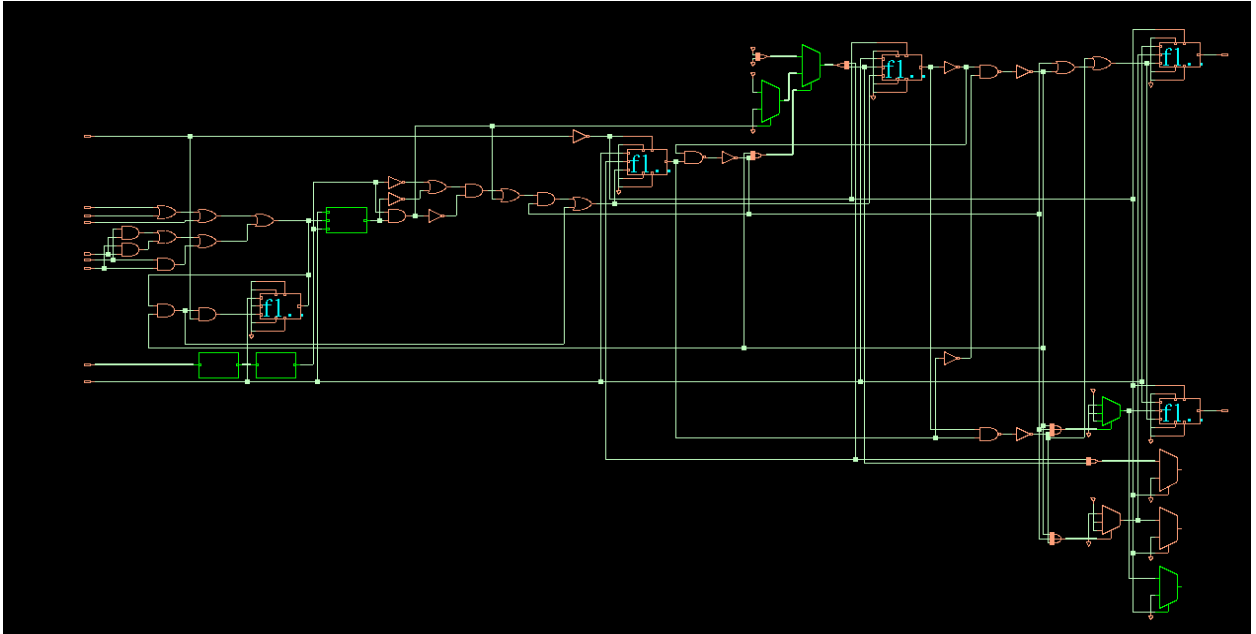


Figure 5: schematic of Verilog design

### 2) Layout:

After finished the design of schematic and synthesis file, it is time to draw the layout by cadence. The process is also based on the previous lab, after open the cadence; it could establish the import design. Then make kinds of setting to get the placement layout, route layout and the layout going through a filter. Meanwhile, at the end of layout design, it is necessary to verify the layout to confirm everything is correct. The verification result will be shown on this lab report.

*a. placement*

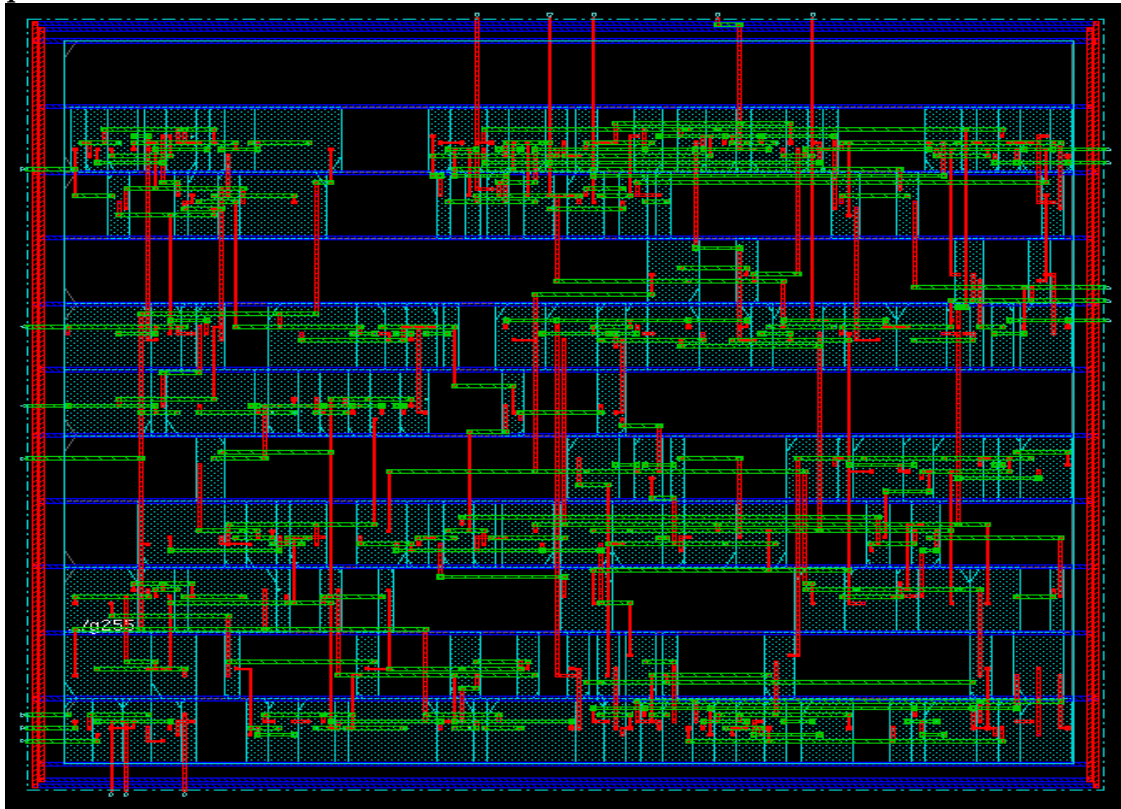


Figure 6: layout for placement

*b. special route*

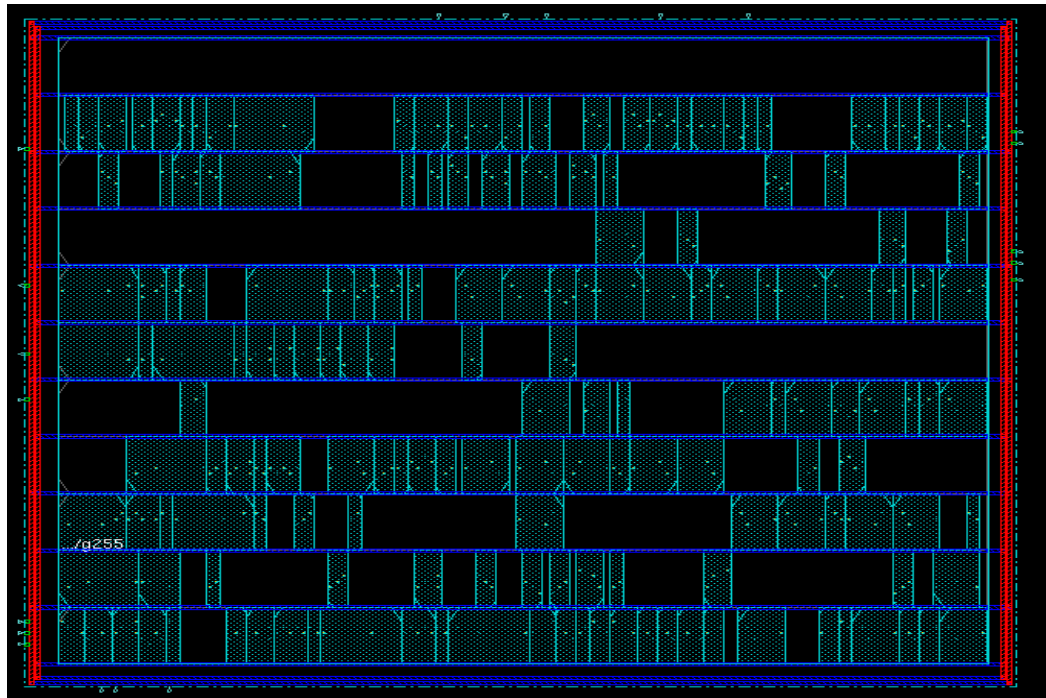


Figure 7: layout for special route

## *c. nanoroute*

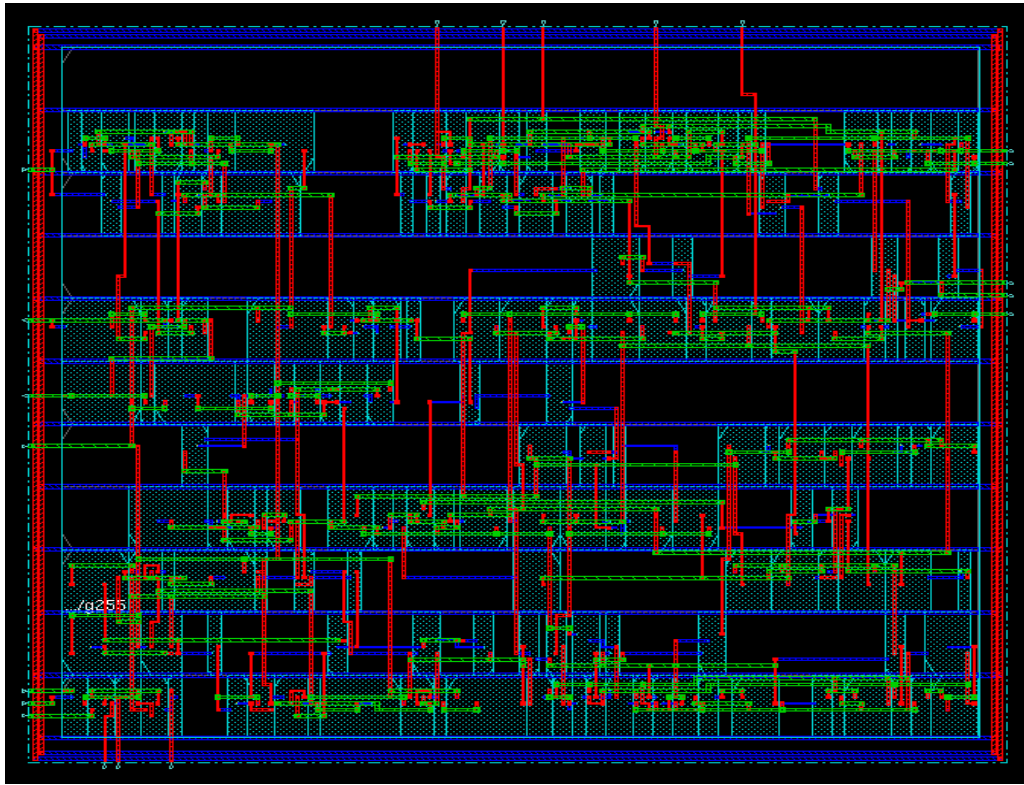


Figure 8: layout for nano route

## *d. Filter Cell*

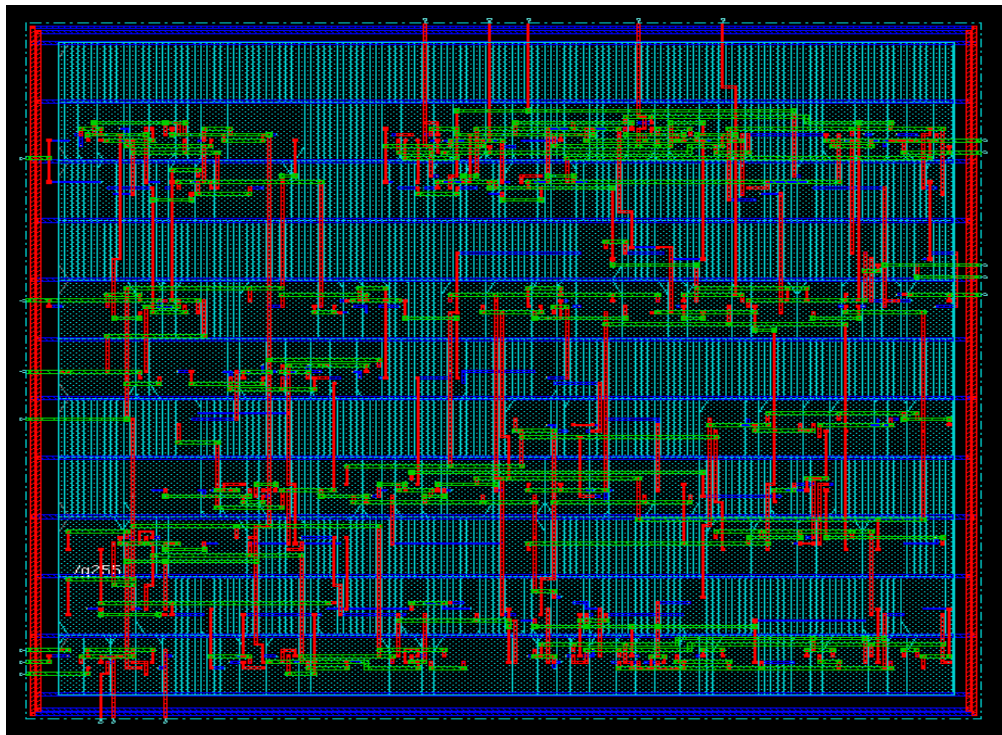


Figure 9: layout after filter cell

## *e. verification:*

### *Geometry verification*

```
encounter 1> *** Starting Verify Geometry (MEM: 243.0) ***

  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                        ..... bin size: 9600
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (ENCVFG-47):   Pin of Cell gate2/g255 at (12.000, 101.100), (36.600, 10
2.900) on Layer metall is not connected to any net. Use globalNetConnect or GUI
Power->Connect Global Nets to specify global net connection rules properly.

  VERIFY GEOMETRY ..... Cells           : 0 Viols.
  VERIFY GEOMETRY ..... SameNet          : 0 Viols.
  VERIFY GEOMETRY ..... Wiring           : 0 Viols.
  VERIFY GEOMETRY ..... Antenna          : 0 Viols.
  VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols. 0 Wrngs.

*****End: VERIFY GEOMETRY*****
*** verify geometry (CPU: 0:00:00.1 MEM: 1.1M)

encounter 1>
```

### *Connectivity verification*

```
encounter 1>
***** Start: VERIFY CONNECTIVITY *****
Start Time: Mon Dec 5 17:44:29 2011

Design Name: BurglarAlarm2
Database Units: 1000
Design Boundary: (0.0000, 0.0000) (353.7000, 352.0500)
Error Limit = 1000; Warning Limit = 50
Check all nets
Time Elapsed: 0:00:00.0

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Mon Dec 5 17:44:29 2011
***** End: VERIFY CONNECTIVITY *****
  Verification Complete : 0 Viols. 0 Wrngs.
  (CPU Time: 0:00:00.0 MEM: 0.031M)
```

*f.. DEF import into Cadence:*

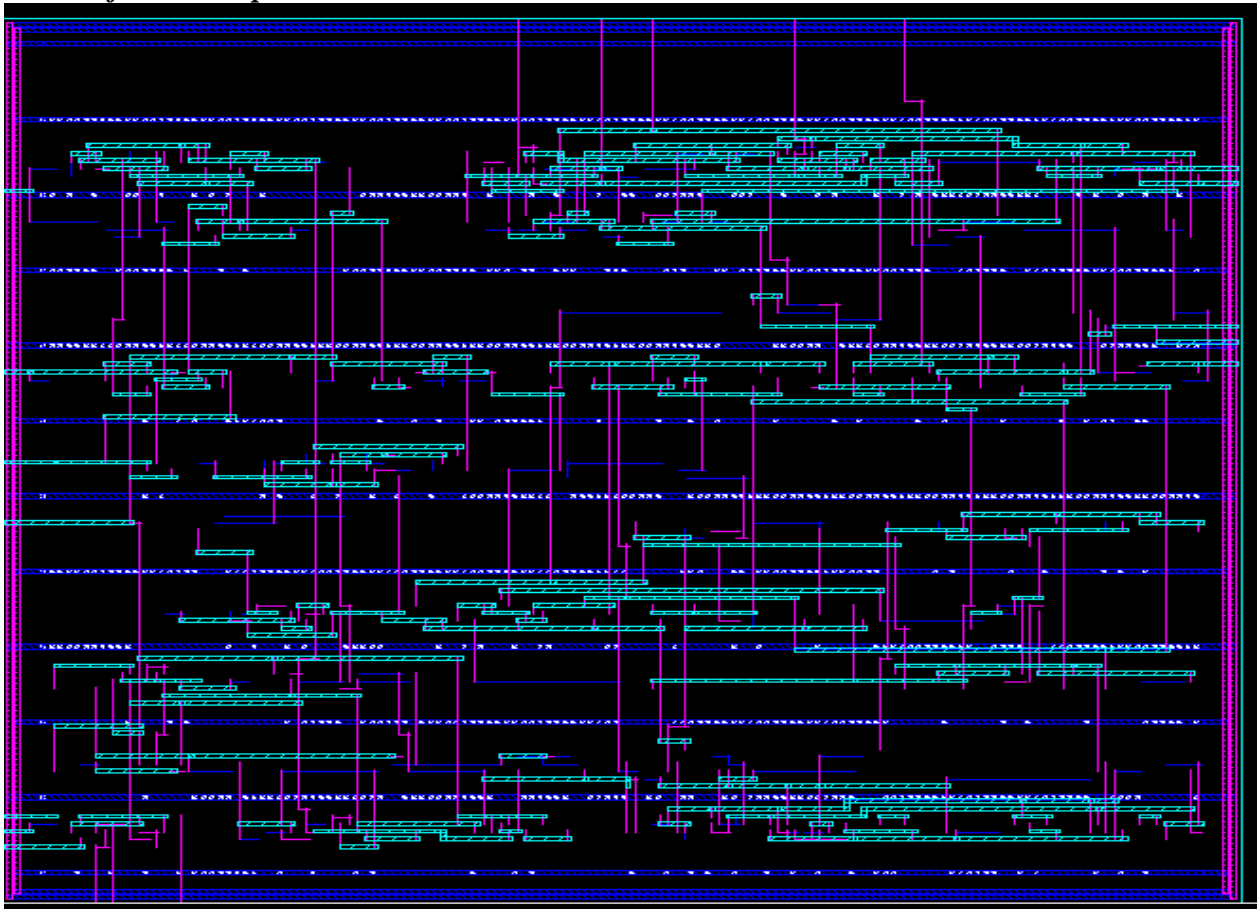


Figure 10: DEF import Cadence

*g. schematic, layout, extracted in Cadence*

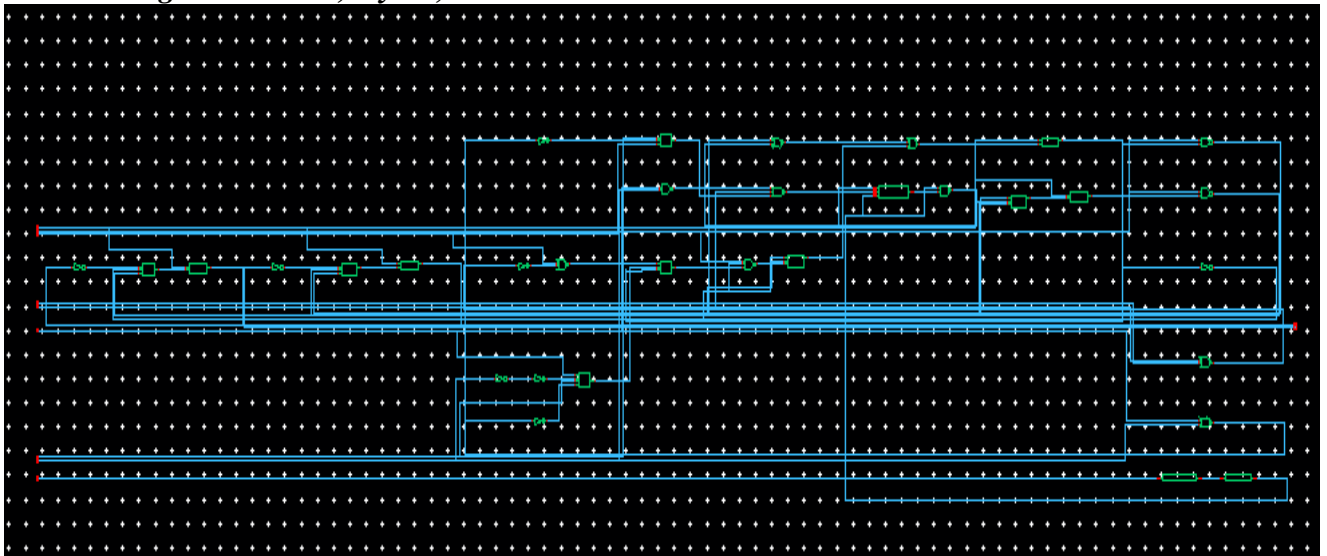


Figure 11:schematic in Cadence



## Layout in Cadence

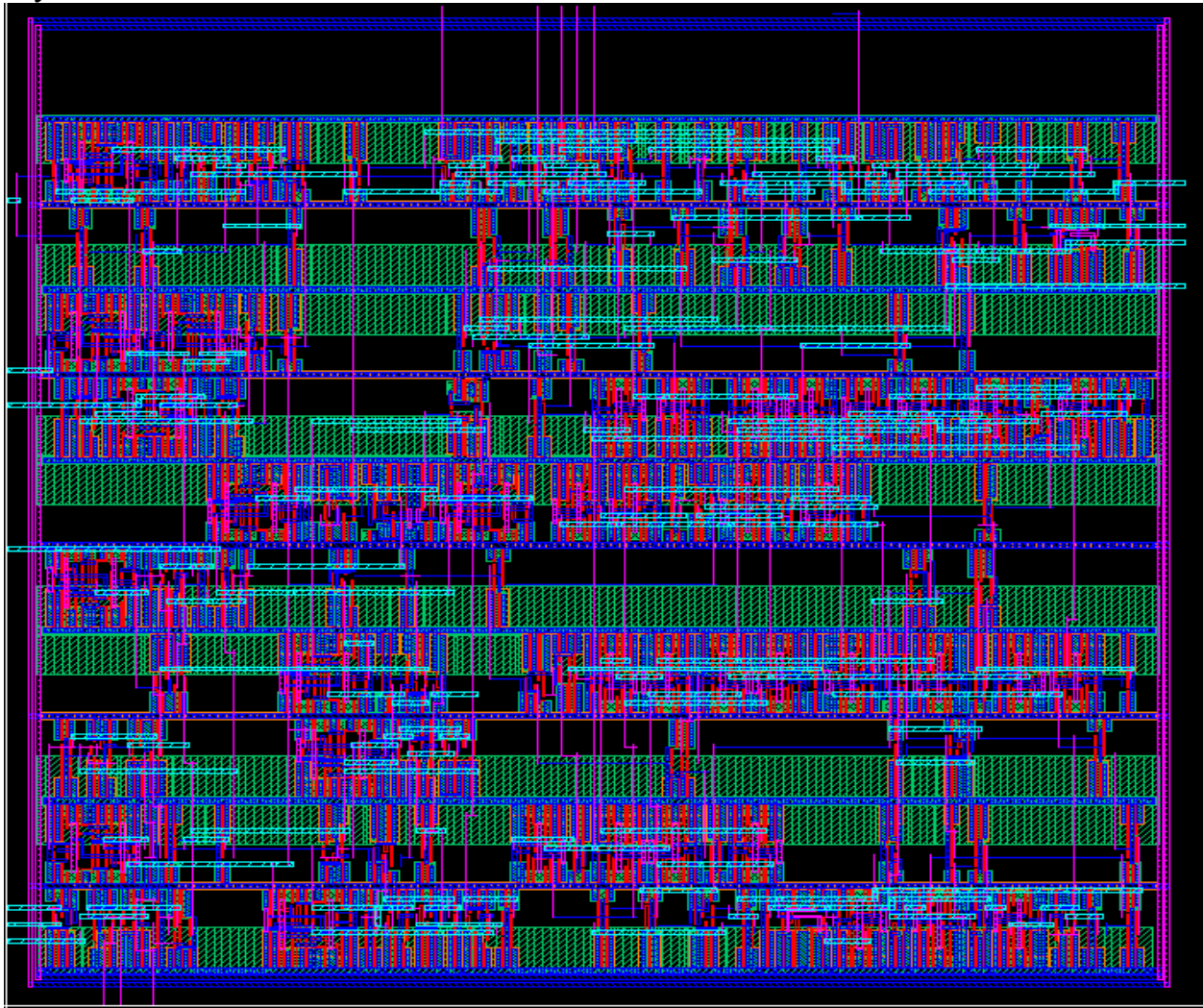


Figure 12: layout in Cadence

## DRC result:

```

drc(elecHighresEdge (notch < (lambda * 7.0)) errMesg)
Executing: drc(highresEdge elecHighresEdge (enc < (lambda * 2.0)) errMesg)
DRC started.....Thu Dec  8 20:49:36 2011
  completed ....Thu Dec  8 20:49:40 2011
  CPU TIME = 00:00:01  TOTAL TIME = 00:00:04
***** Summary of rule violations for cell "BurglarAlarm2 layout" *****
# errors  Violated Rules
      1  (SCMOS Rule 7.2) metall spacing: 0.90 um
      2  dubiousData(("metall" "drawing") "Improperly formed shape - metall")
      9  dubiousData(("metal2" "drawing") "Improperly formed shape - metal2")
     11  dubiousData(("metal3" "drawing") "Improperly formed shape - metal3")
     23  Total errors found
  
```

## Extracted layout in Cadence:

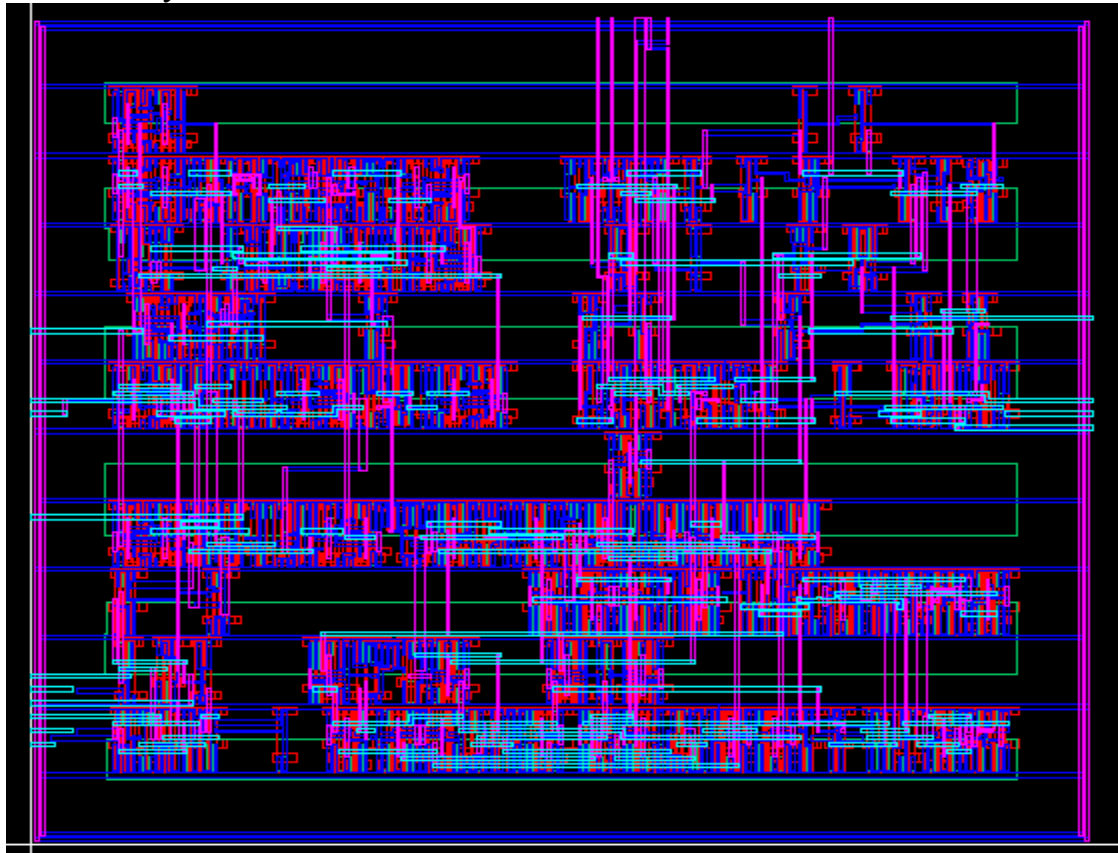


Figure 13: extracted layout

## Result for LVS:

The net-lists failed to match.

	layout	schematic	
		instances	
un-matched	364	460	
rewired		23	0
size errors	0	0	
pruned		0	0
active		1112	1192
total		1112	1192
		nets	
un-matched	249	295	
merged		0	0
pruned		0	0
active		629	668
total		629	668
		terminals	
un-matched	0	0	
matched but			
different type		0	0
total		0	22

## ***Analysis:***

In this final project, the Verilog part is successful because the result is exactly same with the expected ones. However, after import the schematic and layout into Cadence, it is really hard to make them match each other; the miss matched parts are so many. By trying many methods and looking for many help, this part is still not easy to check all of them and revised them.

## ***Conclusion:***

This final project is about design a burglar alarm by Verilog to satisfy the requirement of this system function. In the whole design process, the biggest problem is to figure out how to implement each module and connect them together. Meanwhile, there are some coding rules which are not exactly same with C language or Java language. In the coding process, there are some troubles to find out the precise method to write the code. As for the schematic and synthesis part, that is relative smoother than Verilog, even though there are some error needed to revised based on the synthesis principles. Overall, this project is not hard as imaged and it is a good understanding to Verilog and digital circuit.