

## PART 1: DIGITAL DESIGN THEORY

Provide your answers here:

1.

I 8 Base number 10 ~ 32:

10 11 12 13 14 15 16 17 20 21 22 23 24 25 26 27 30 31 32 33 34 35 36 37 40

II 16 Base number 10~32

A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1F 20

III 12 Base number 10~32

A B 10 11 12 13 14 15 16 17 18 19 1A 1B 20 21 22 23 24 25 26 27 28

2. The largest number can be expressed with 12 bits is  $111111111111_{(2)}$ , in 10 base in it is  $8191_{(10)}$

3.

I: transfer 512 from base 10 to base 2,  $512\%2 = 0$ , so the final number is 0,  $512/2 = 256$ ,  $256\%2 = 0$ , so the second last number is 0,  $256/2 = 128$ ,  $128\%2 = 0$ , so the third last number is 0,  $128/2 = 64$ ,  $64\%2 = 0$ , so the fourth last number is 0,  $64/2 = 32$ ,  $32\%2 = 0$ , so the fifth last number is 0,  $32/2 = 16$ ,  $16\%2 = 0$ , so the sixth last number is 0,  $16/2 = 8$ ,  $8\%2 = 0$ , so the seven last number is 0,  $8/2 = 4$ ,  $4\%2 = 0$ , so the eighth last number is 0,  $4/2 = 2$ ,  $2\%2 = 0$ , so the ninth last number is 0,  $2/2 = 1 < 2$ , so the first one is 1. So it is  $1000000000$

II:  $512\%16 = 0$ , So the final number is 0,  $512/16 = 32$ ,  $32\%16 = 0$ , so the second last number is 0,  $32/16 = 2 < 16$ , so the first number is 2, and the  $512_{(10)}$  in 16 base is  $200_{(16)}$ , and  $2_{(16)}$  in 2 base is  $0010$ ,  $0_{(16)}$  in 2 base is  $0000$ , and it is  $001000000000$ , delete the zero before 2, it is  $1000000000_{(2)}$

III Compare the two way, it is obviously that the second way is more easy. The first one take 9 steps and second way just take  $3+3 = 6$  steps.

4.

Three basic type of logic calculation

I : And

Input1	Input2	Output
0	0	0
0	1	0
1	0	0
1	1	1

II : Or

Input1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	1

III: No

Input	Output
0	1
1	0

5. the code of the following decimal number

I 7214564 and the 9 scale inverse code is 7214563

II 36764994 the 9 scale inverse code is 36764993

## PART 2: DIGITAL DESIGN LAB (TASK1)

### DESIGN

The addition of my system is first define two input value length 2 to accept the value from switch,two output variable length 2 to output the value input. And a variable length3 to be the sum of the two input values

Code :

```
`timescale 1ns / 1ps
```

```
module addition(
```

```
    input [1:0]A,
```

```
    input [1:0]B,
```

```
    output[2:0]C,
```

```
    output[1:0]D,
```

```
    output[1:0]E
```

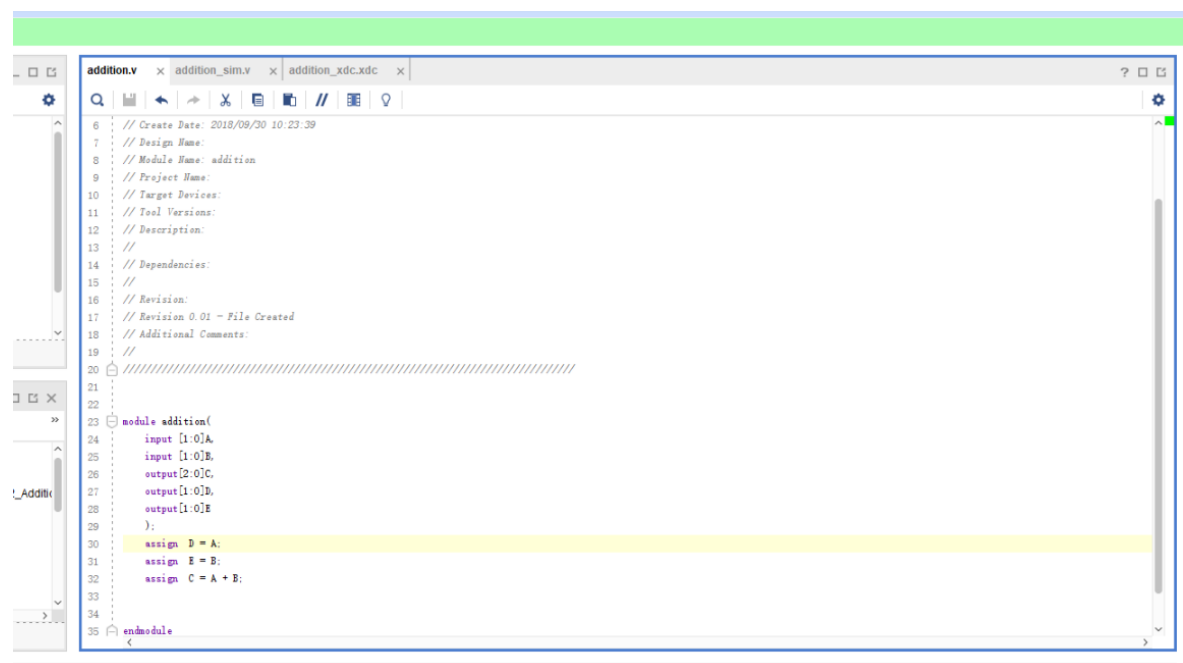
```
);
```

```
    assign  D = A;
```

```
    assign  E = B;
```

```
    assign  C = A + B;
```

```
endmodule
```



## SIMULATION

---

The simulation document first define reg AB and wire C,D,E

And then make the value a and b change in regular

Code:

```
`timescale 1ns / 1ps

module addition_sim(

    );

    reg A,B;

    wire C,D,E;

    addition u_duf(

        .A(A),

        .B(B),

        .C(C),

        .D(D),

        .E(E)

    );


    initial

    begin

        A = 0; B =0;

        #10;

        A = 0; B =1;

        #10;

        A = 1; B =0;

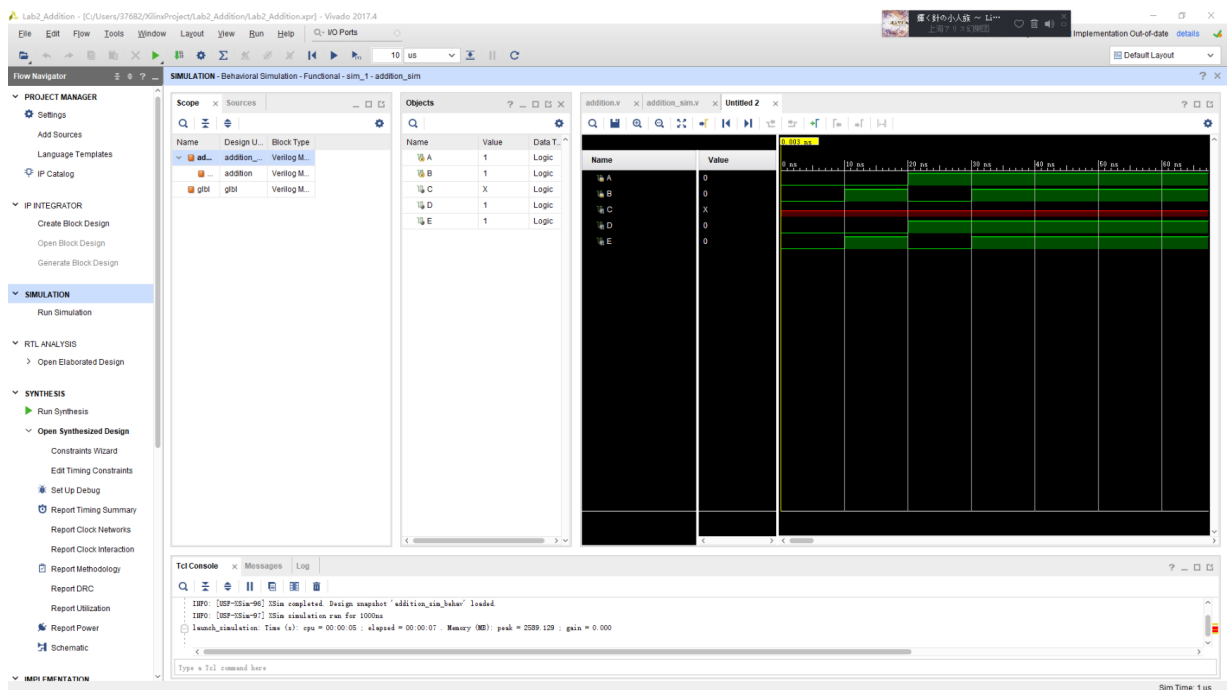
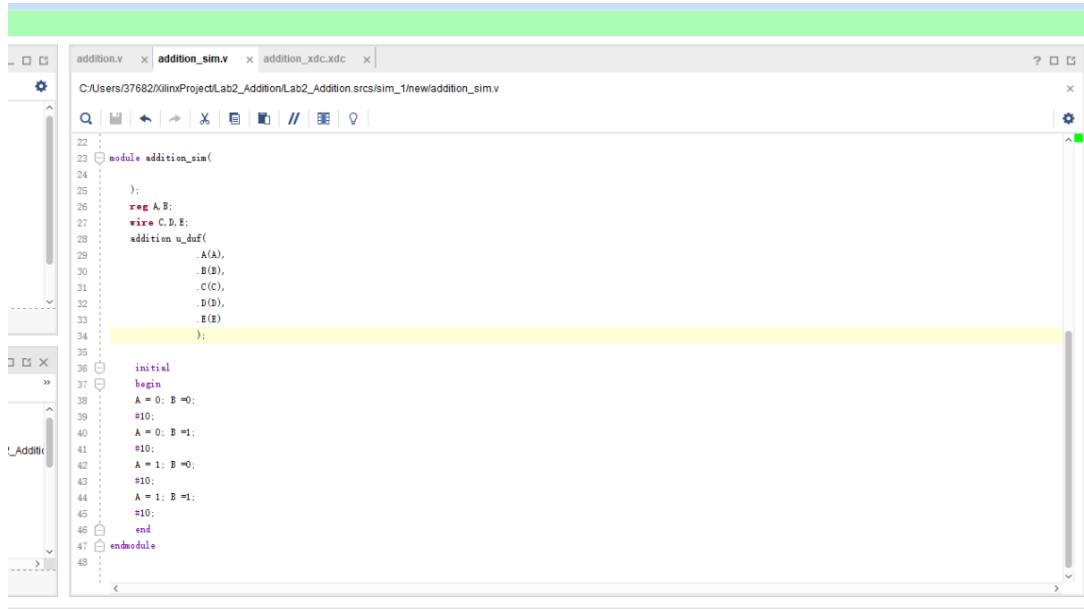
        #10;
```

A = 1; B = 1;

#10;

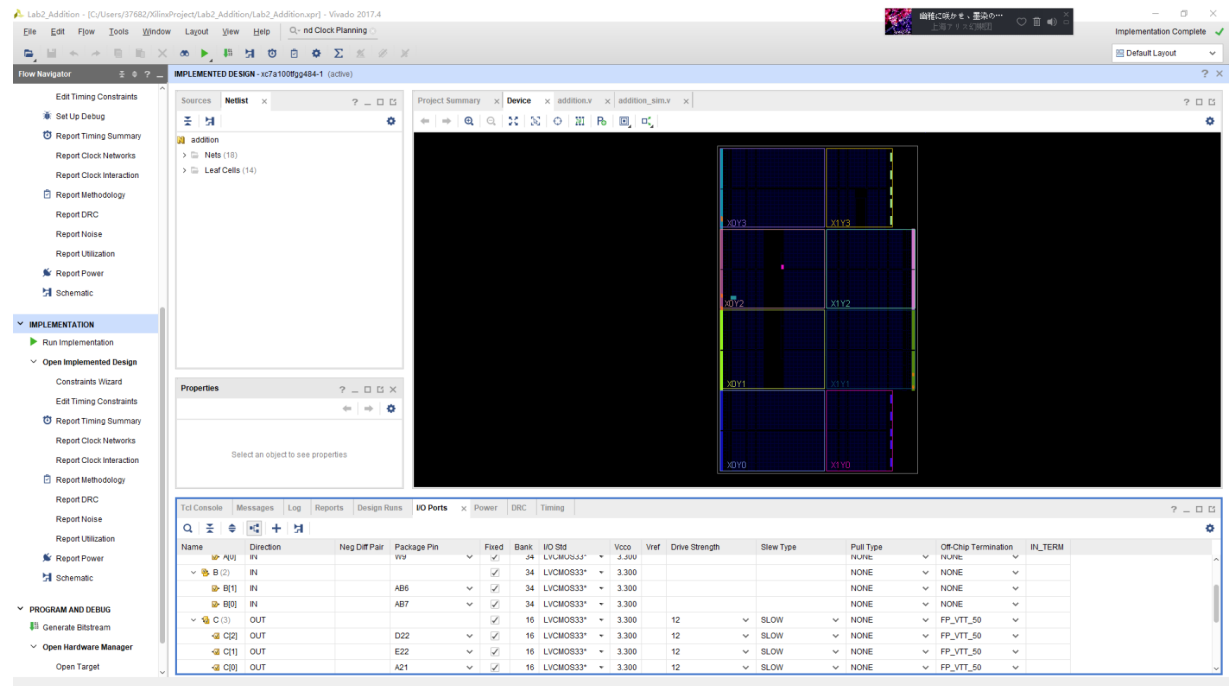
end

endmodule



## CONSTRAINT FILE AND THE TESTING

Then build the xdc file using the i/o port tools



It will produce a xdc file

Code:

```
set_property PACKAGE_PIN Y9 [get_ports {A[1]}]
set_property PACKAGE_PIN W9 [get_ports {A[0]}]
set_property DRIVE 12 [get_ports {C[2]}]
set_property DRIVE 12 [get_ports {C[1]}]
set_property DRIVE 12 [get_ports {C[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {C[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {C[1]}]
```

```

set_property IOSTANDARD LVCMOS33 [get_ports {C[0]}]

set_property PACKAGE_PIN AB6 [get_ports {B[1]}]

set_property PACKAGE_PIN AB7 [get_ports {B[0]}]

set_property PACKAGE_PIN D22 [get_ports {C[2]}]

set_property PACKAGE_PIN E22 [get_ports {C[1]}]

set_property PACKAGE_PIN A21 [get_ports {C[0]}]

set_property PACKAGE_PIN K17 [get_ports {D[1]}]

set_property PACKAGE_PIN L13 [get_ports {D[0]}]

set_property PACKAGE_PIN M17 [get_ports {E[1]}]

set_property PACKAGE_PIN M16 [get_ports {E[0]}]

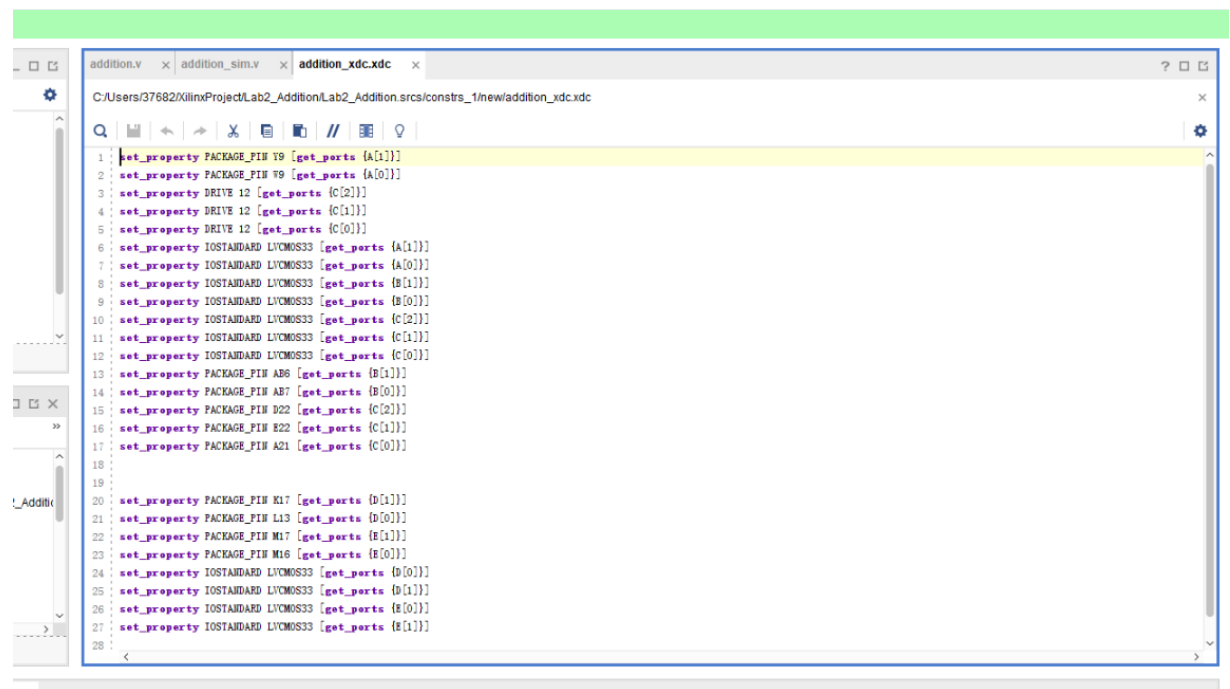
set_property IOSTANDARD LVCMOS33 [get_ports {D[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {D[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {E[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {E[1]}]

```

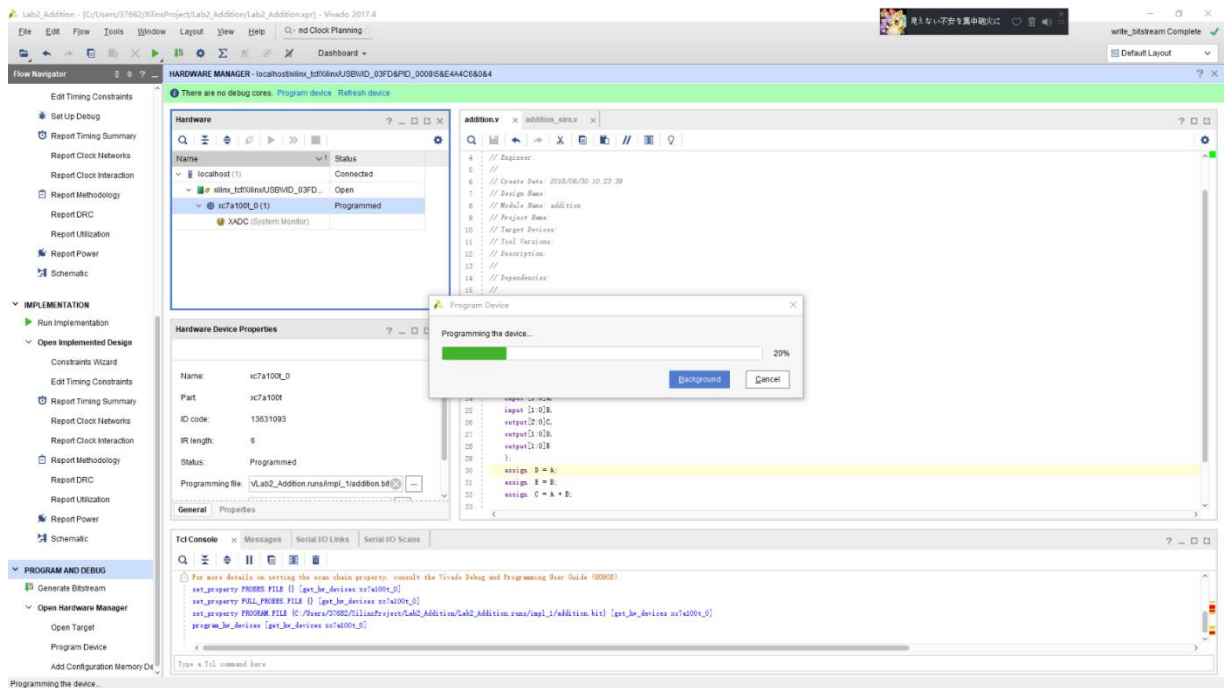


```

1 set_property PACKAGE_PIN Y9 [get_ports {A[1]}]
2 set_property PACKAGE_PIN Y9 [get_ports {A[0]}]
3 set_property DRIVE 12 [get_ports {C[2]}]
4 set_property DRIVE 12 [get_ports {C[1]}]
5 set_property DRIVE 12 [get_ports {C[0]}]
6 set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
7 set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
8 set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
9 set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
10 set_property IOSTANDARD LVCMOS33 [get_ports {C[2]}]
11 set_property IOSTANDARD LVCMOS33 [get_ports {C[1]}]
12 set_property IOSTANDARD LVCMOS33 [get_ports {C[0]}]
13 set_property PACKAGE_PIN AB6 [get_ports {B[1]}]
14 set_property PACKAGE_PIN AB7 [get_ports {B[0]}]
15 set_property PACKAGE_PIN D22 [get_ports {C[2]}]
16 set_property PACKAGE_PIN E22 [get_ports {C[1]}]
17 set_property PACKAGE_PIN A21 [get_ports {C[0]}]
18
19
20 set_property PACKAGE_PIN K17 [get_ports {D[1]}]
21 set_property PACKAGE_PIN L13 [get_ports {D[0]}]
22 set_property PACKAGE_PIN M17 [get_ports {E[1]}]
23 set_property PACKAGE_PIN M16 [get_ports {E[0]}]
24 set_property IOSTANDARD LVCMOS33 [get_ports {D[0]}]
25 set_property IOSTANDARD LVCMOS33 [get_ports {D[1]}]
26 set_property IOSTANDARD LVCMOS33 [get_ports {E[0]}]
27 set_property IOSTANDARD LVCMOS33 [get_ports {E[1]}]
28

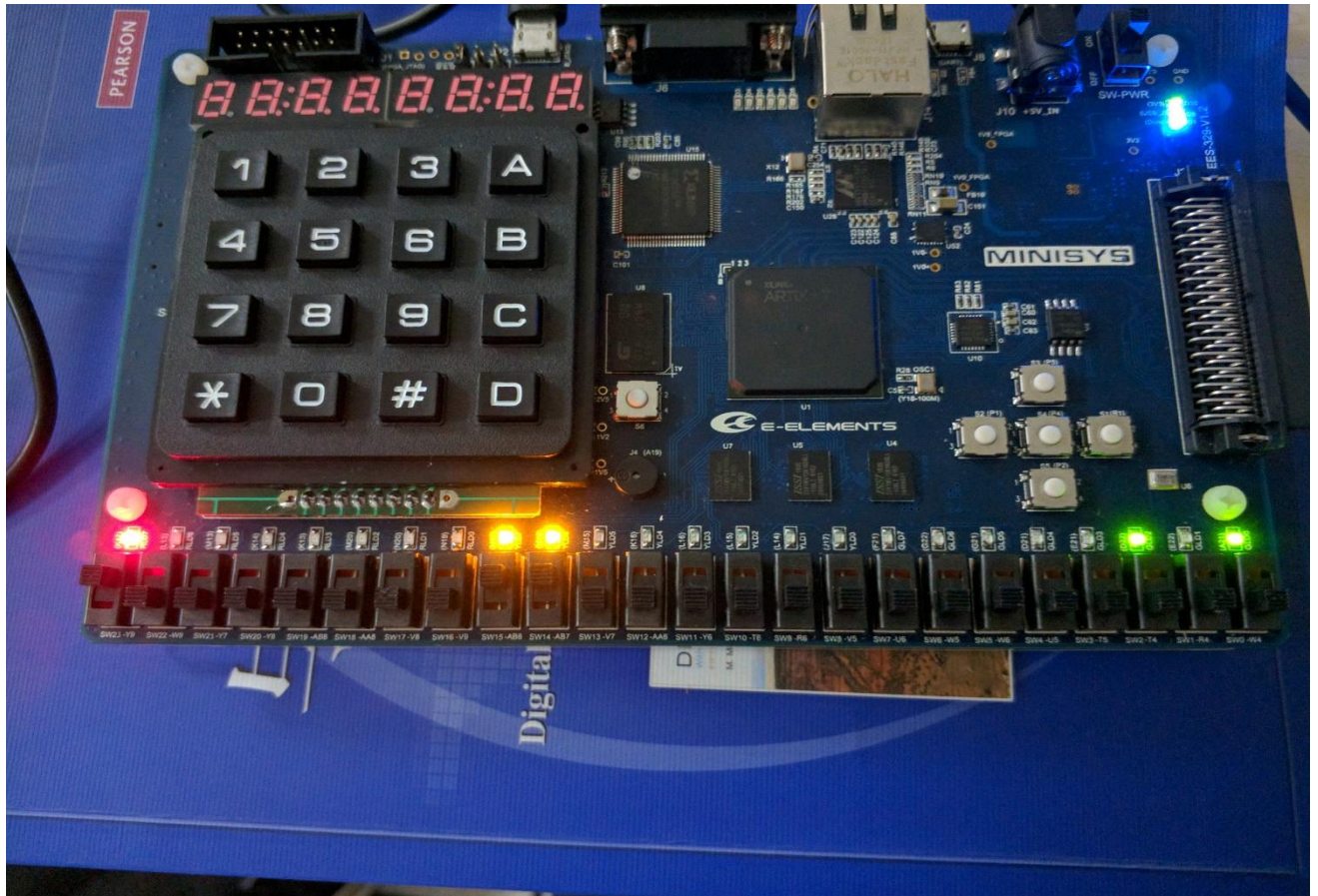
```

after that produce the bitstream file to burn it the the Minisys Practice platform.

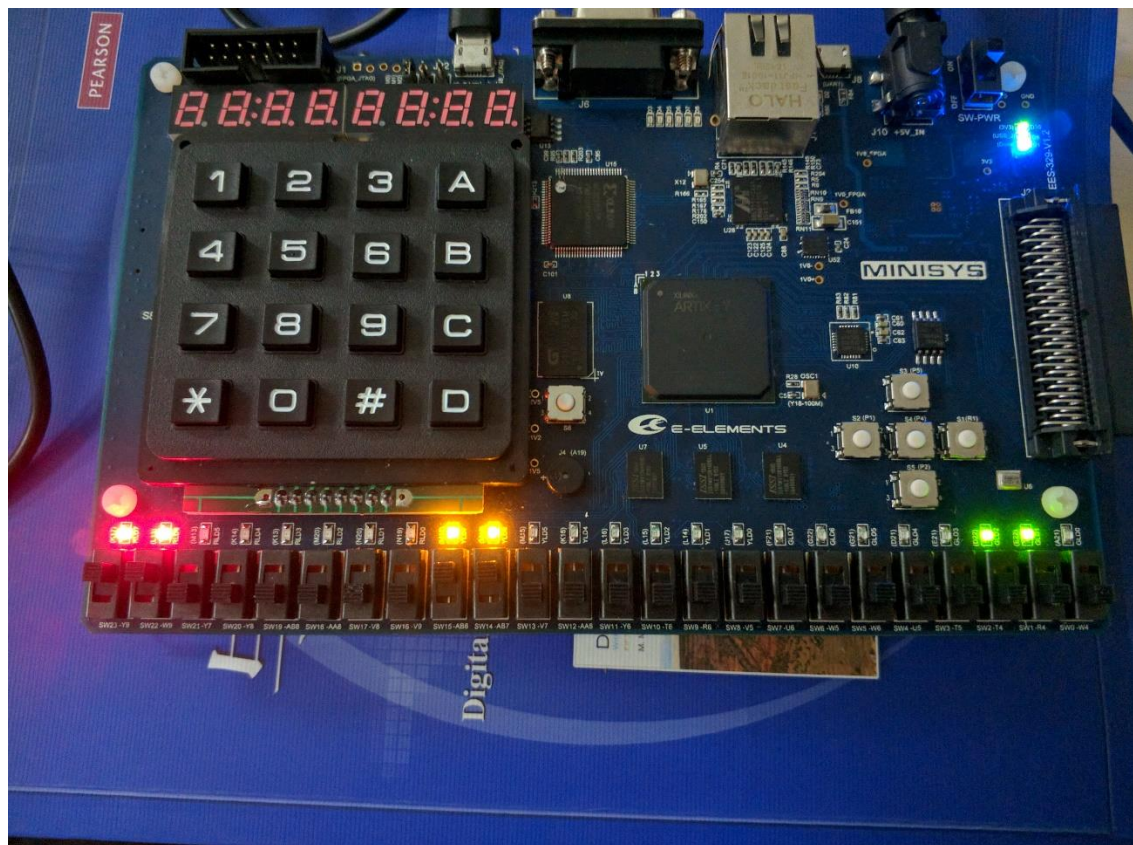


Using the green button to auto connect and then point the program device ,program will be burn in the platform





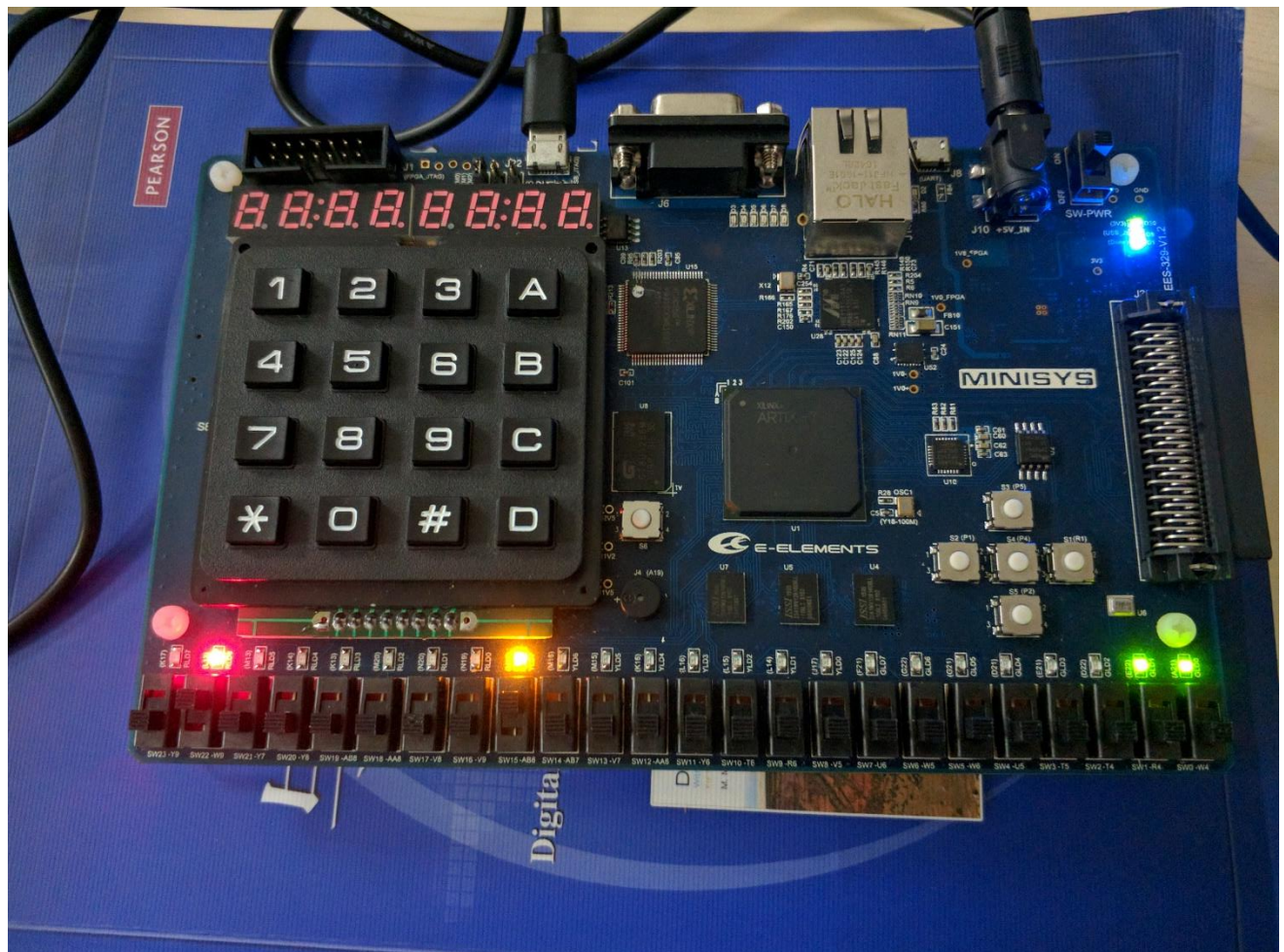
$$Y9W9(10) + AB6AB7(11) = T4R4W4 (101)$$





$Y9W9(11) + AB6AB7(11) = T4R4W4(110)$

$//K17L13(11) + M17M16(11) = D22E22A21(101)$



$Y9W9(01) + AB6AB7(10) = T4R4W4(011)$

## THE DESCRIPTION OF OPERATION

I build a new project and named it then add source or later choose the artix7fgg100-1's type

II then we got in the real surface ,we can add a file in the design source and named it with two input and three output , then we count them in the film with assign prefix then we add a simulation file without any input or output but define reg and wire variable then bind the input with reg and output with wire. then it will change every #10; to change form 0 and 1. Finally before the endmole needs a end.

III do the simulation and do the synthesis will be Smooth sailing, after simulation we will get the simulation graph like ladder. And the synthesis will bring a graph like real chips

IV Do the implementation, after that we can define the i/o ports we set input with the switch's code and output with their code at the order. Do not forget the voltage to 3.3V

V produce the bitstream file and connect the platform. If nothing had been found ,connect the usb once again and click it to stop server. Then do auto connect

VI Finally program the device in the green strip, we will have a programed platform, test it and take photos.

## PART 2: DIGITAL DESIGN LAB (TASK2)

### DESIGN

---

I the data flow way have two input and six output. Two of the output is to produce the switches' condition and the others is the answer of the count.

Code:

```
`timescale 1ns / 1ps
```

```
module demorgan_df(
```

```
    input input1,
```

```
    input input2,
```

```
    output outinput1,
```

```
    output outinput2,
```

```
    output output1,
```

```
    output output2,
```

```
    output output3,
```

```
    output output4
```

```
);
```

```
    assign outinput1 = input1;
```

assign outinput2 = input2;

assign output1 = ~(input1 | input2);

assign output2 = (~input1)&(~input2);

assign output3 = ~(input1 & input2);

assign output4 = (~input1)|(~input2);

endmodule

```

C:/Users/37682/XilinxProject/demorgan/demorgan.srcs/sources_1/new/demorgan_df.v
1 // Tool Versions:
2 // Description:
3 //
4 // Dependencies:
5 //
6 // Revision:
7 // Revision 0.01 - File Created
8 // Additional Comments:
9 //
0 ///////////////////////////////////////////////////////////////////
1
2
3 module demorgan_df(
4     input input1,
5     input input2,
6     output outinput1,
7     output outinput2,
8     output output1,
9     output output2,
0     output output3,
1     output output4
2 );
3     assign outinput1 = input1;
4     assign outinput2 = input2;
5     assign output1 = ~(input1 | input2);
6     assign output2 = (~input1)&(~input2);
7     assign output3 = ~(input1 & input2);
8     assign output4 = (~input1)|(~input2);
9
0 endmodule
1
```

II Structured design using the graph way to design it and this is the graph and it's code is

Code:

```
`timescale 1ns / 1ps

module demorgan_sd(

    input input1,

    input input2,

    output outinput1,

    output outinput2,

    output output1,

    output output2,

    output output3,

    output output4

);

    assign outinput1 = input1;

    assign outinput2 = input2;

    wire temp0;

    orgate_0 u_or_0(

        .a(input1),

        .b(input2),

        .q(temp0)

    );

    notgate_0 u_not_0(
```

```

        .a(temp0),

        .c(output1)

    );

    wire temp1;

    wire temp2;

    notgate_0 u_not_1(

        .a(input1),

        .c(temp1)

    );

    notgate_0 u_not_2(

        .a(input2),

        .c(temp2)

    );

    andgate_0 u_add_0(

        .a(temp1),

        .b(temp2),

        .q(output2));

    orgate_0 u_or_1(

        .a(temp1),

        .b(temp2),

        .q(output4)

    );

```

```
wire temp3;

andgate_0 u_add_1(

    .a(input1),

    .b(input2),

    .q(temp3)

);

notgate_0 u_not_3(

    .a(temp3),

    .c(output3)

);

endmodule
```

```

`timescale 1ns / 1ps

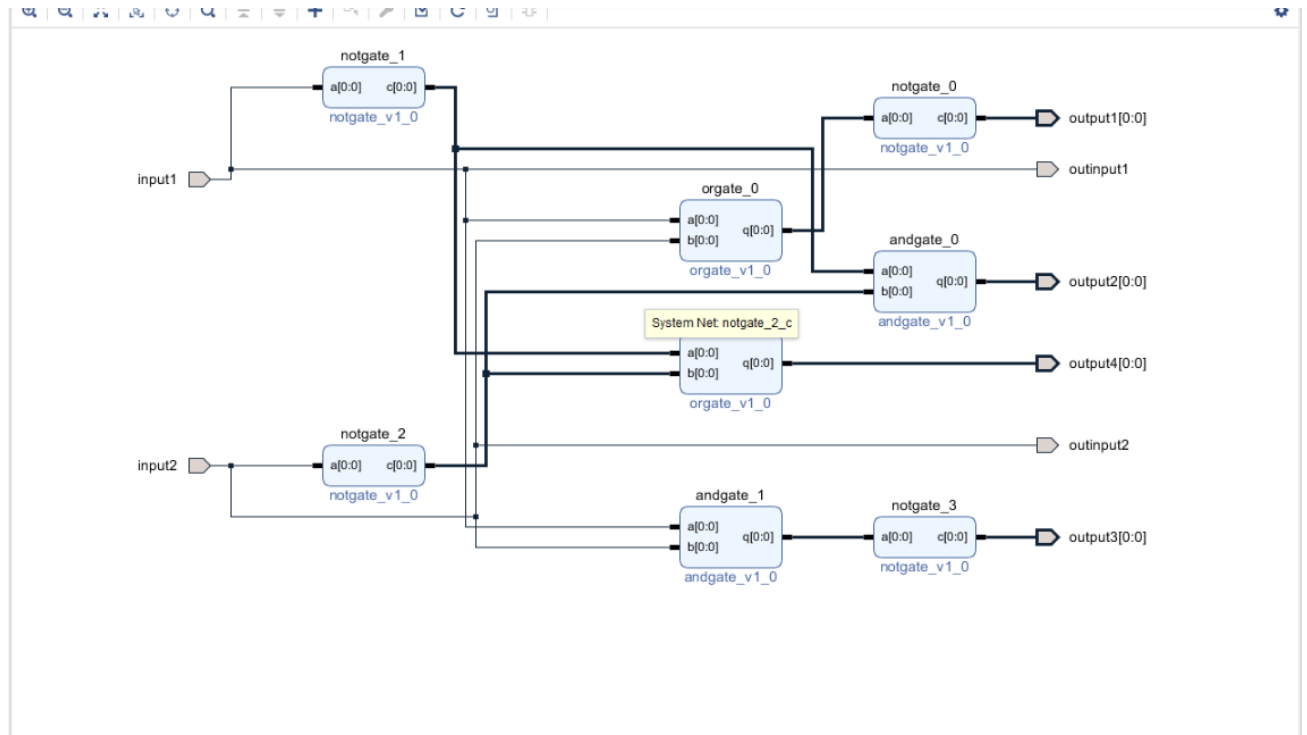
module demorgan_sd(
    input input1,
    input input2,
    output outinput1,
    output outinput2,
    output output1,
    output output2,
    output output3,
    output output4

);
    assign outinput1 = input1;
    assign outinput2 = input2;
    wire temp0;
    orgate_0 u_or0(
        .a(input1),
        .b(input2),
        .q(temp0)
    );
    notgate_0 u_not_0(
        .a(temp0),
        .c(output1)
    );
    wire temp1;
    wire temp2;
    notgate_0 u_not_1(
        .a(input1),
        .c(temp1)
    );
    wire temp2;
    notgate_0 u_not_1(
        .a(input1),
        .c(temp1)
    );
    notgate_0 u_not_2(
        .a(input2),
        .c(temp2)
    );
    andgate_0 u_add0(
        .a(temp1),
        .b(temp2),
        .q(output2)
    );
    wire temp3;
    andgate_0 u_add1(
        .a(input1),
        .b(input2),
        .q(temp3)
    );
    notgate_0 u_not_3(
        .a(temp3),
        .c(output3)
    );
    orgate_0 u_or1(
        .a(temp1),
        .b(temp2),
        .q(output4)
    );
endmodule

```

III the third way is the block design





## SIMULATION

I Code:

```
`timescale 1ns / 1ps
```

```
module demorgan_sim();
```

```
    reg siminput1,siminput2;
```

```
    wire simoutput1,simoutput2,simoutput3,simoutput4;
```

```
    demorgan_df demo_df(
```

```
        .input1(siminput1),
```

```
        .input2(siminput2),
```

```
        .output1(simoutput1),
```

```
        .output2(simoutput2),
```

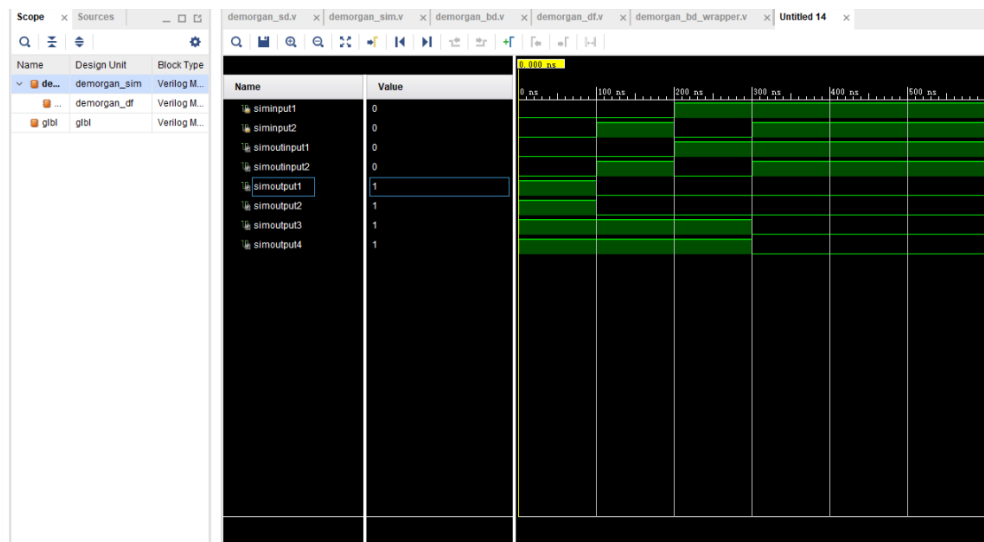
```
.output1(simoutput1),  
  
.output2(simoutput2),  
  
.output3(simoutput3),  
  
.output4(simoutput4)  
  
    );  
  
initial  
  
begin  
  
    siminput1=0;  
  
    siminput2=0;  
  
    #100  
  
    siminput1=0;  
  
    siminput2=1;  
  
    #100  
  
    siminput1=1;  
  
    siminput2=0;  
  
    #100  
  
    siminput1=1;  
  
    siminput2=1;  
  
end  
  
endmodule
```

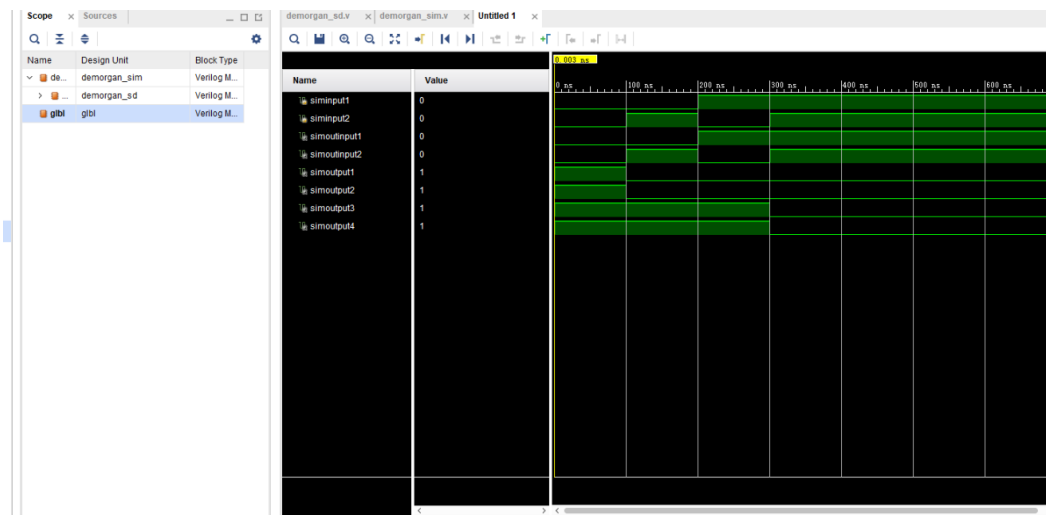
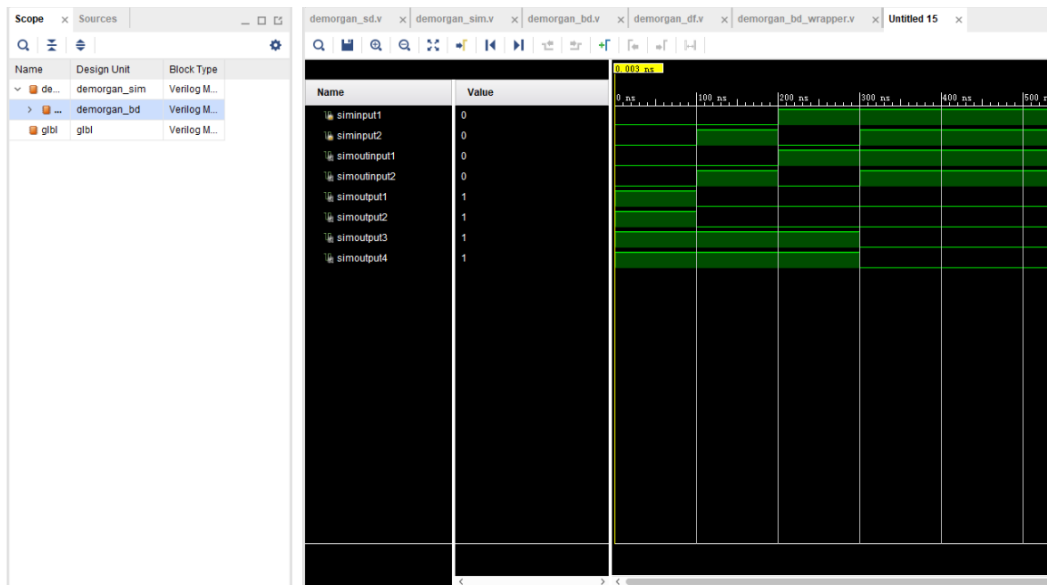
```

    }
    module demorgan_sim();
        reg siminput1, siminput2;
        wire simoutinput1, simoutinput2, simoutput1, simoutput2, simoutput3, simoutput4;
        demorgan_df demo_df(
            .input1(siminput1),
            .input2(siminput2),
            .outinput1(simoutinput1),
            .outinput2(simoutinput2),
            .output1(simoutput1),
            .output2(simoutput2),
            .output3(simoutput3),
            .output4(simoutput4)
        );
    initial
    begin
        siminput1=0;
        siminput2=0;
        #100
        siminput1=0;
        siminput2=1;
        #100
        siminput1=1;
        siminput2=0;
        #100
        siminput1=1;
        siminput2=1;
    end
endmodule

```

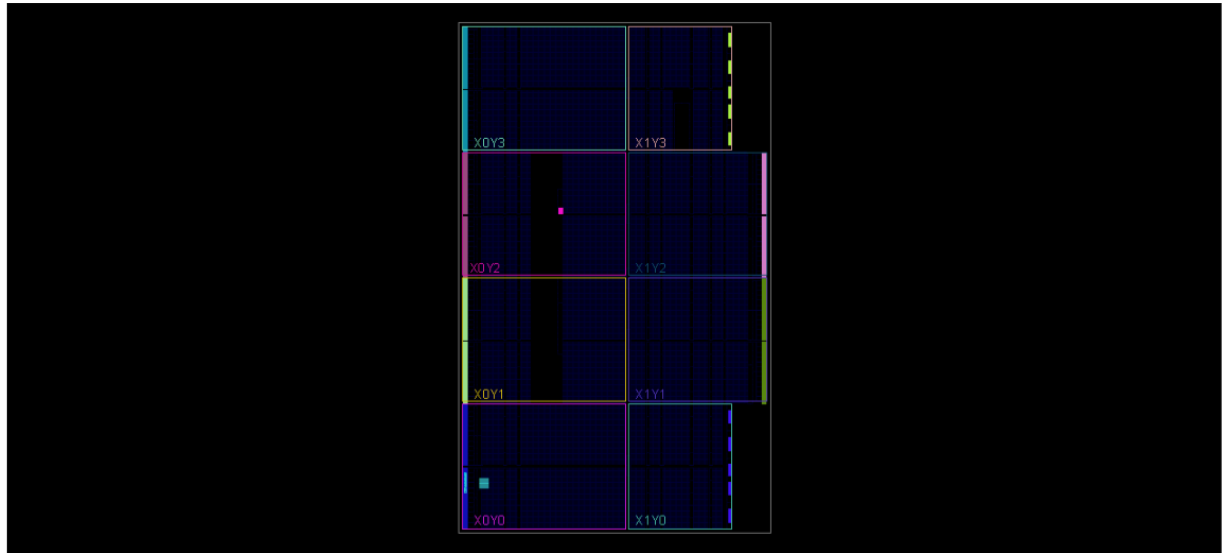
If the three pictures are different from the left words





## CONSTRAINT FILE AND THE TESTING

All ports (8)													
Scalar ports (4)													
input1	IN		Y9	✓	34	LVC MOS33*	3.300				NONE	NONE	
input2	IN		W9	✓	34	LVC MOS33*	3.300				NONE	NONE	
oufinput1	OUT		K17	✓	15	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
oufinput2	OUT		L13	✓	15	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
output1 (1)	OUT			✓	16	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
output1[0]	OUT		G22	✓	16	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
output2 (1)	OUT			✓	16	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
output2[0]	OUT		D21	✓	16	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
output3 (1)	OUT			✓	16	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
output3[0]	OUT		D22	✓	16	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
output4 (1)	OUT			✓	16	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		
output4[0]	OUT		A21	✓	16	LVC MOS33*	3.300	12	SLOW	NONE	FP_VTT_50		



and the xdc files is

Codes:

```
set_property PACKAGE_PIN G22 [get_ports {output1[0]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {output1[0]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {output2[0]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {output3[0]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {output4[0]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports input1]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports input2]
```

set\_property IOSTANDARD LVCMOS33 [get\_ports outinput1]

set\_property IOSTANDARD LVCMOS33 [get\_ports outinput2]

set\_property PACKAGE\_PIN D22 [get\_ports {output3[0]}]

set\_property PACKAGE\_PIN A21 [get\_ports {output4[0]}]

set\_property PACKAGE\_PIN Y9 [get\_ports input1]

set\_property PACKAGE\_PIN W9 [get\_ports input2]

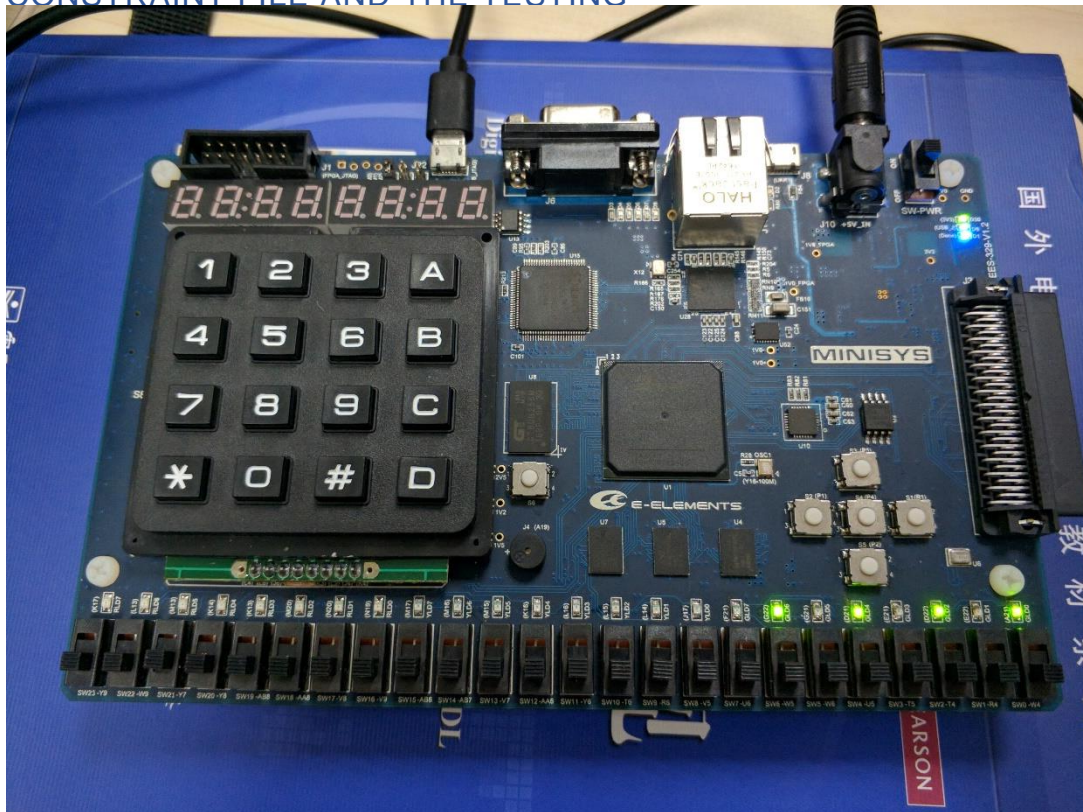
set\_property PACKAGE\_PIN K17 [get\_ports outinput1]

set\_property PACKAGE\_PIN L13 [get\_ports outinput2]

set\_property PACKAGE\_PIN D21 [get\_ports {output2[0]}]

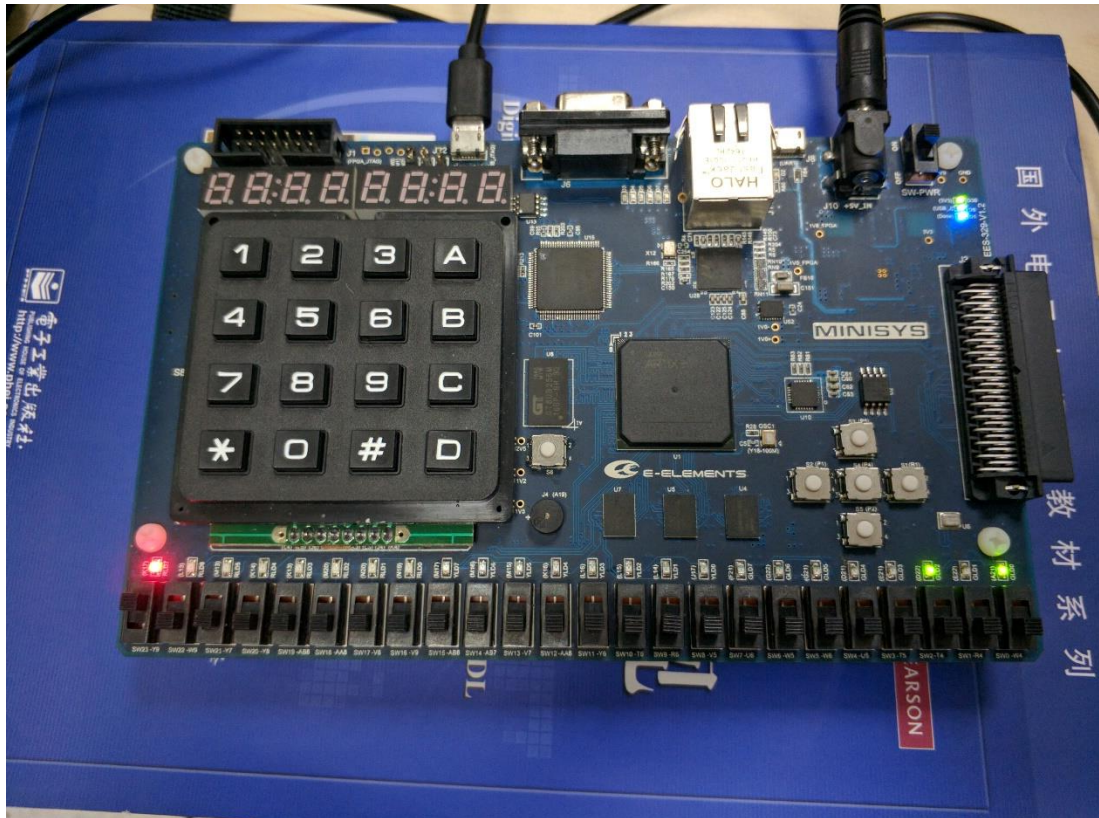
```
set_property PACKAGE_PIN G22 [get_ports {output1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output3[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output4[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports input1]
set_property IOSTANDARD LVCMOS33 [get_ports input2]
set_property IOSTANDARD LVCMOS33 [get_ports outinput1]
set_property IOSTANDARD LVCMOS33 [get_ports outinput2]
set_property PACKAGE_PIN D22 [get_ports {output3[0]}]
set_property PACKAGE_PIN A21 [get_ports {output4[0]}]
set_property PACKAGE_PIN Y9 [get_ports input1]
set_property PACKAGE_PIN W9 [get_ports input2]
set_property PACKAGE_PIN K17 [get_ports outinput1]
set_property PACKAGE_PIN L13 [get_ports outinput2]
set_property PACKAGE_PIN D21 [get_ports {output2[0]}]
```

## CONSTRAINT FILE AND THE TESTING

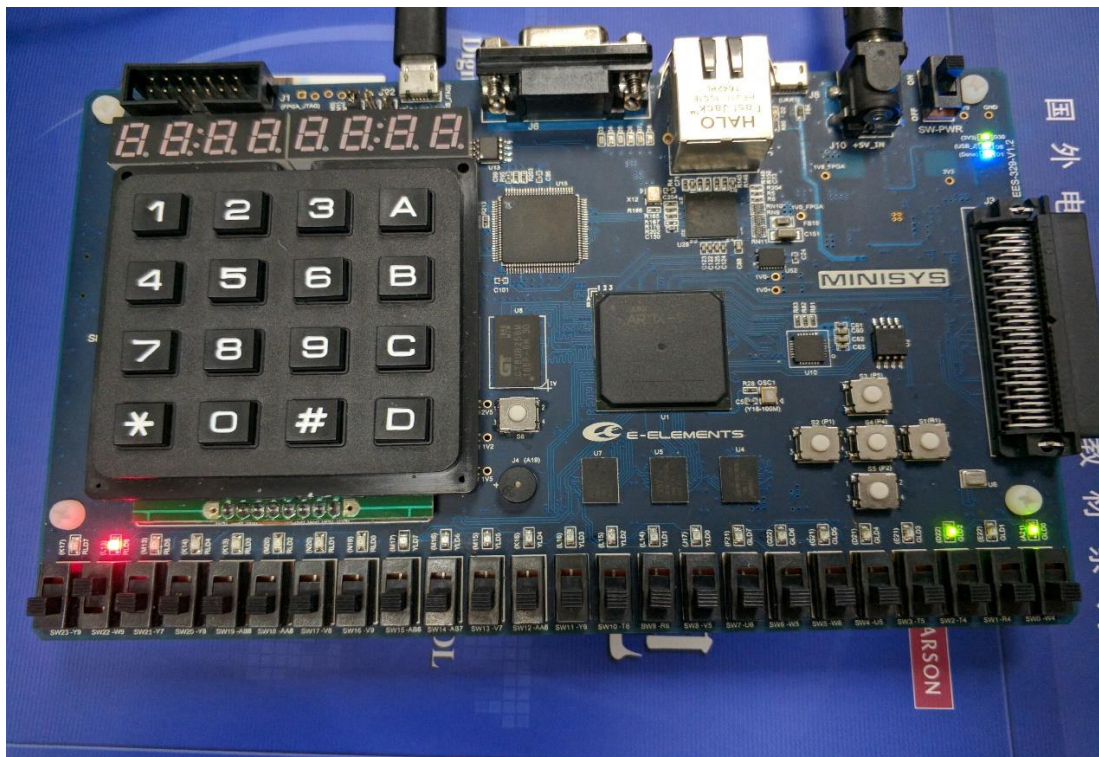


$$Y9(0) + W9(0) \rightarrow W5(1) + U5(1) + T4(1) + W4(1)$$



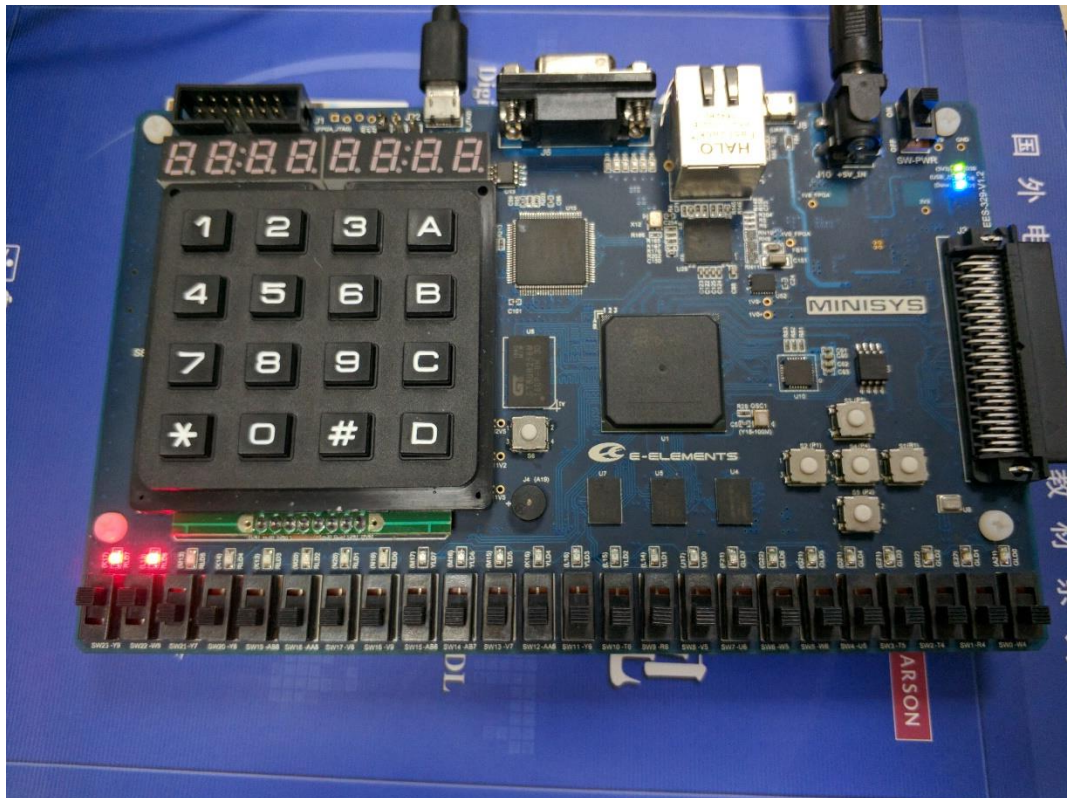


$Y9(1) + W9(0) \rightarrow W5(0) + U5(0) + T4(1) + W4(1)$



$Y9(0) + W9(1) \rightarrow W5(0) + U5(0) + T4(1) + W4(1)$





$$Y9(1) + W9(1) \rightarrow W5(0) + U5(0) + T4(0) + W4(0)$$

It is obviously that the W5&U5, T4&W4 this double couple have the same condition so the equation is right

## THE DESCRIPTION OF OPERATION

I build a new project and named it then add source or later choose the artix7fgg100-1's type

II then we got in the real surface ,we can add a file in the design source and named it with two input and six output , then we count them in the film with assign prefix. Then we add the ips in our files and add them in our demorgan\_sd to design it .after all we using the fuction in the and or and not gates to produce and bd files.

III then we add a simulation file without any input or output but define reg and wire variable then bind the input with reg and output with wire. then it will change every #10; to change form 0 and 1. Finally before the endmole needs a end. And we will make III IV and V steps three times to use every file we build .

III do the simulation and do the synthesis will be Smooth sailing, after simulation we will get the simulation graph like ladder. And the synthesis will bring a graph like real chips Sometimes it will happen cannot find ports on this module then we can close vivado and open it once again it will disappear.

IV Do the implementation, after that we can define the i/o ports we set input with the switch's code and output with their code at the order. Do not forget the voltage to 3.3V

V produce the bitstream file and connect the platform. If nothing had been found ,connect the usb once again and click it to stop server. Then do auto connect

VI Finally program the device in the green strip, we will have a programed platform, test it and take photos.