

**DIGITAL DESIGN**

**ASSIGNMENTREPORT**

**ASSIGNMENT ID: 01**

**Student Name: Weibao Fu (傅伟堡)**

**Student ID: 11812202**

PART 1: DIGITAL design THEORY

1.

|  |  |  |  |
| --- | --- | --- | --- |
| Decimal | Octal | Hexadecimal | Base 14 |
| 12 | 14 | C | C |
| 13 | 15 | D | D |
| 14 | 16 | E | 10 |
| 15 | 17 | F | 11 |
| 16 | 20 | 10 | 12 |
| 17 | 21 | 11 | 13 |
| 18 | 22 | 12 | 14 |
| 19 | 23 | 13 | 15 |
| 20 | 24 | 14 | 16 |
| 21 | 25 | 15 | 17 |
| 22 | 26 | 16 | 18 |
| 23 | 27 | 17 | 19 |
| 24 | 30 | 18 | 1A |
| 25 | 31 | 19 | 1B |
| 26 | 32 | 1A | 1C |
| 27 | 33 | 1B | 1D |
| 28 | 34 | 1C | 20 |

2.The largest signed binary number with 14 bits is (01111111111111)2, corresponding to the decimal number is 8191, corresponding to the hexadecimal number is (1FFF)16.

The largest unsigned binary number with 14 bits is (11111111111111)2, corresponding to the decimal number is 16383, corresponding to the hexadecimal number is (3FFF)16.

3.

Convert 240 to binary number:

240/2=120……0 120/2=60……0 60/2=30……0 30/2=15……0

15/2=7……1 7/2=3……1 3/2=1……1 1/2=0……1

Such that the binary number is (11110000)2

Convert 240 to hexadecimal number: 240/16=15……0 15/16=0……15

Such that the hexadecimal number is (F0)16

Convert (F0)16 to binary number (11110000)2

The second method is faster.

4.

AND • : Only when all things happen, the result is true, else the result is false.

|  |  |  |
| --- | --- | --- |
| A | B | A • B |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR + : If One of the things happens, the result is true. Only if all of the things does not happen, the result is false.

|  |  |  |
| --- | --- | --- |
| A | B | A + B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

NOT ’ : When the thing happens, the result is false. When the thing does not happen, the result is true.

|  |  |
| --- | --- |
| A | A’ |
| 0 | 1 |
| 1 | 0 |

5.

14274836/16=892177……4 892177/16=55761……1 55761/16=3485……1

3485/16=217……13 217/16=13……9 13/16=0……13

Such that the decimal number 14274836 corresponds to the hexadecimal number (00D9D114)16. Since the number is positive, the complement code is also (00D9D114)16.

Since the decimal number is negative, transfer it to the binary number (1000 0001 1010 1001 0000 0100 1110 0000)2. Then we can transfer it to Radix-minus-one complement (1111 1110 0101 0110 1111 1011 0001 1111)2, corresponding to (FE56FB1F)16. Then we can get the complement code in binary number (1111 1110 0101 0110 1111 1011 0010 0000)2.Such that the complement code in hexadecimal notation (FE56FB20)16

PART 2: DIGITAL design LAB (Task1)

##### Design

*Verilog Design Code:*

module add\_design(addend, augend, addend\_led, augend\_led, sum\_led);  
input [1:0]addend;  
input augend;  
output [1:0]addend\_led;  
output augend\_led;  
output [2:0]sum\_led;  
assign addend\_led=addend;  
assign augend\_led=augend;  
assign sum\_led = addend+augend;  
endmodule

*Truth Table:*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Add1 | Add2 | Aug | Add | Aug | Sum1 | Sum2 | Sum3 | Sum |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 2 |
| 1 | 1 | 0 | 3 | 0 | 0 | 1 | 1 | 3 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 2 |
| 1 | 0 | 1 | 2 | 1 | 0 | 1 | 1 | 3 |
| 1 | 1 | 1 | 3 | 1 | 1 | 0 | 0 | 4 |

##### simulation

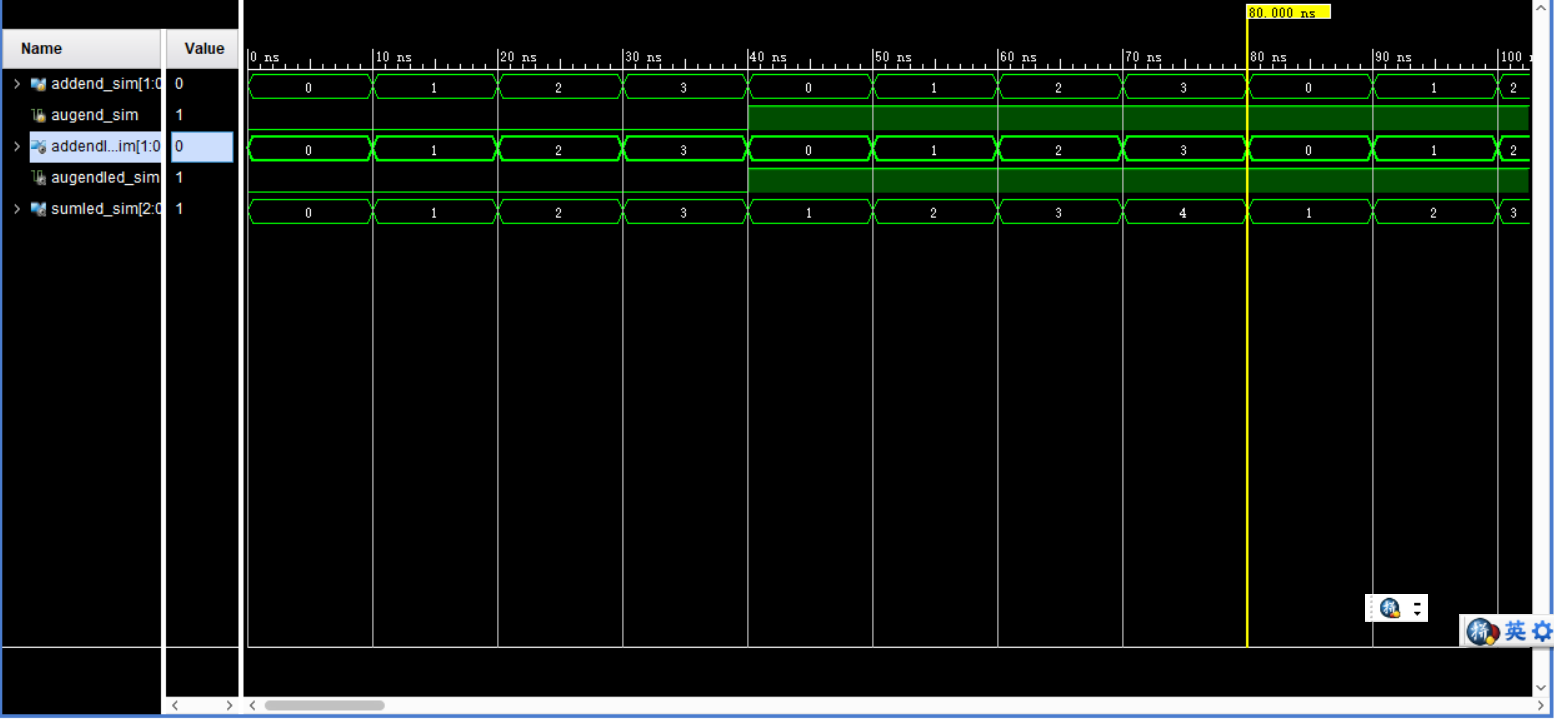
*Verilog Simulation Code:*

module add\_sim( );  
reg [1:0]addend\_sim;  
reg augend\_sim;  
wire [1:0]addendled\_sim;  
wire augendled\_sim;  
wire [2:0]sumled\_sim;  
​  
add\_design u(  
.addend(addend\_sim),.augend(augend\_sim),.addend\_led(addendled\_sim),.augend\_led(augendled\_sim),.sum\_led(sumled\_sim));

initial  
begin  
   addend\_sim=0;augend\_sim=0;  
   #40 augend\_sim=1;  
end

always  
begin  
    #10 addend\_sim = addend\_sim+1;  
end  
​  
endmodule

*Wave Form:*



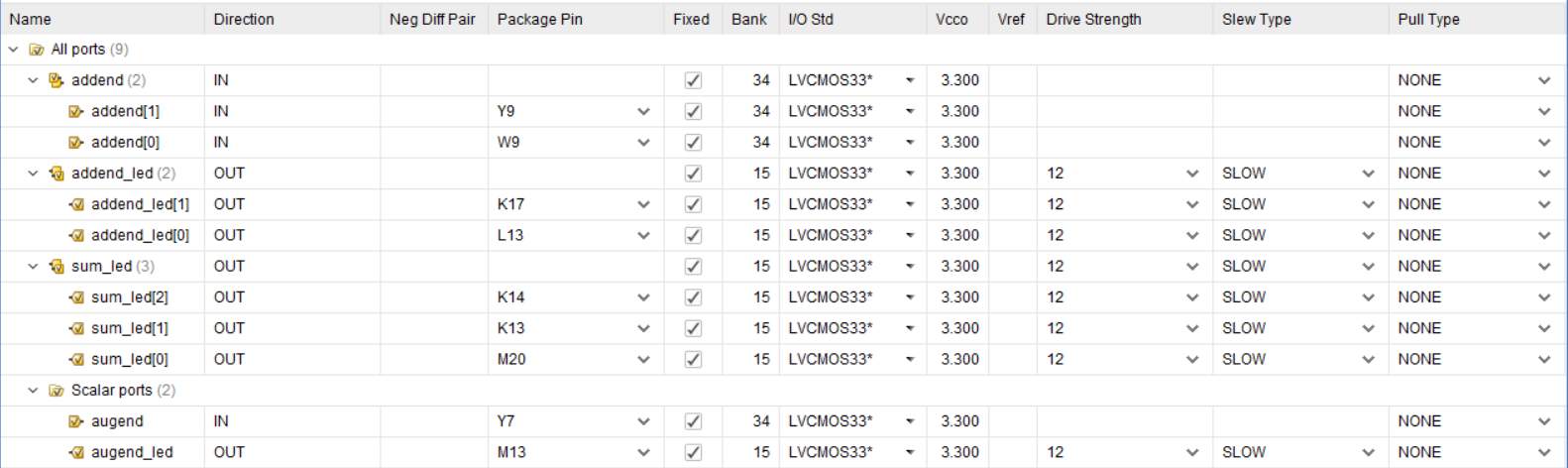
*Analysis*

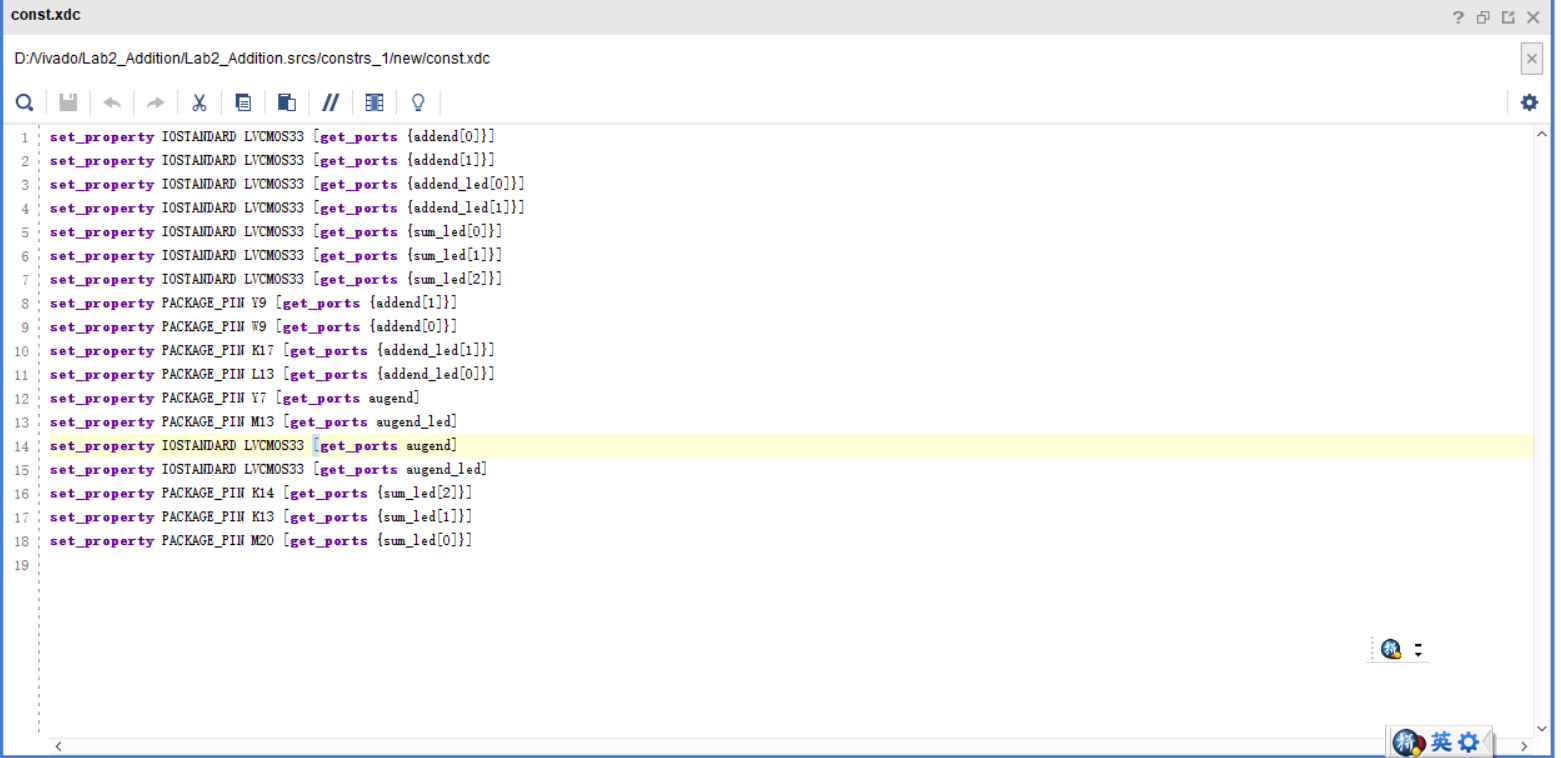
*When 0ns – 40ns, augend is equal to 0, and let addend add 1 per 10ns from 0. It shows that the sum also add 1 per 10 ns from 0, which is same as the truth table.*

*When 40ns – 80ns, augend is equal to 1, and let addend add 1 per 10ns from 0. It shows that the sum add 1 per 10 ns from 1, which is also same as the truth table.*

*Above all, the simulation result is same as the truth table and the function of design meets our expectation.*

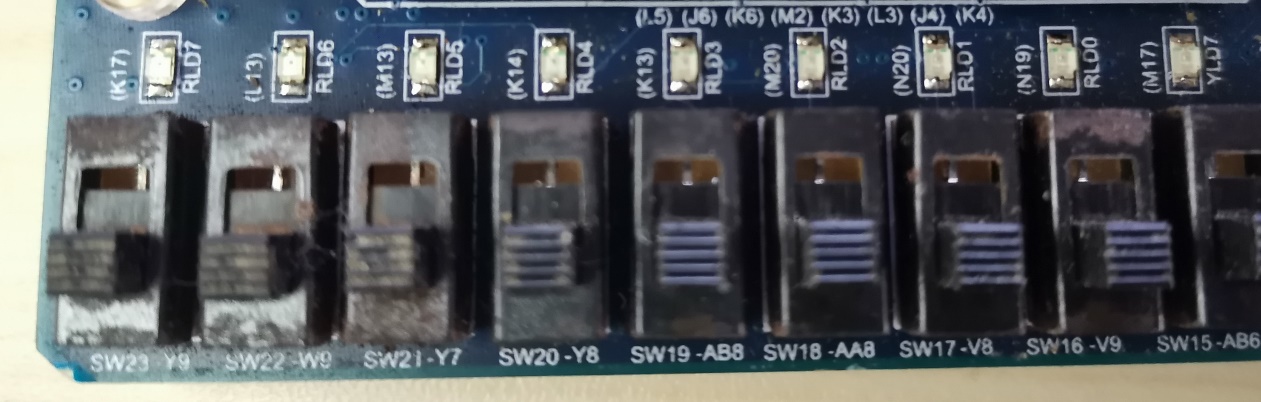
##### Constraint file and the testing

*Constraint File* 

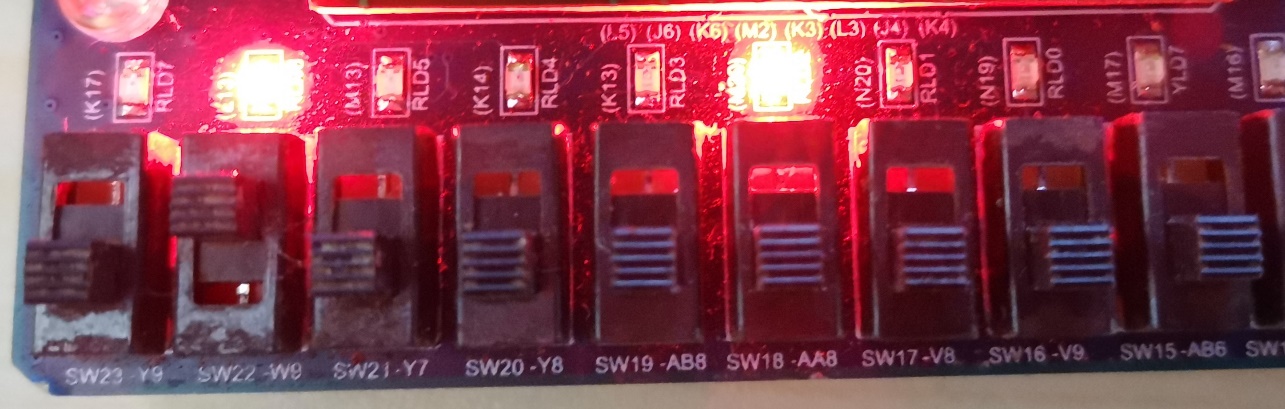


*Testing:*

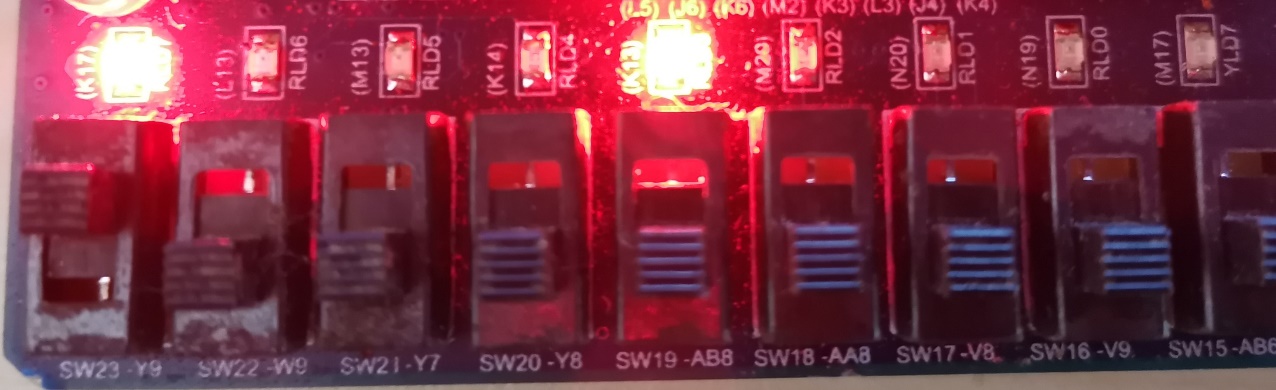
*Step1: set augend(Y7) off(augend=0), and addend be 0(both Y9,W9 off), then the result is below.(K14,K13,M20 are all off, which mean the result is (000)2, corresponding to 0)*

**

*Step2: set augend(Y7) off(augend=0), and addend be 1(Y9 off,W9 on), then the result is below.(K14,K13 are off and M20 is on, which mean the result is (001)2, corresponding to 1)*

**

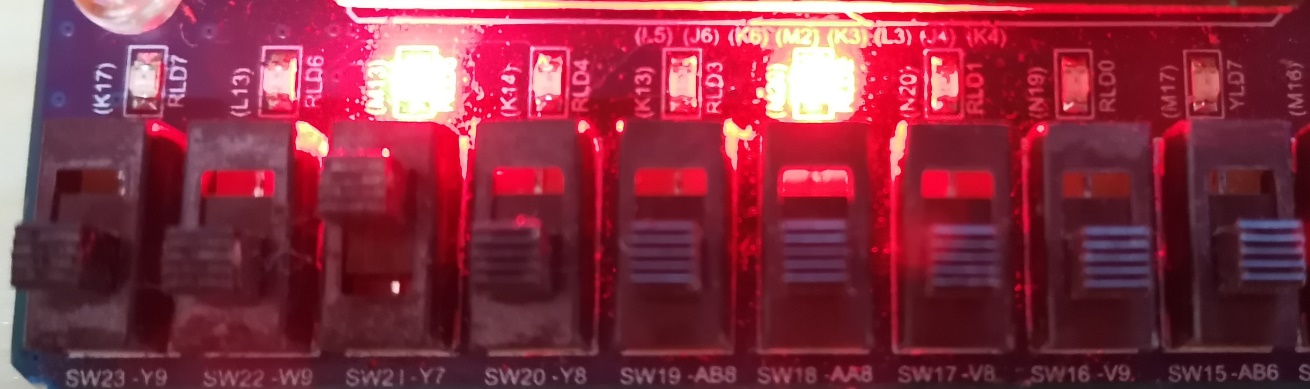
*Step3: set augend(Y7) off(augend=0), and addend be 2(Y9 on,W9 off), then the result is below.(K14,M20 are off and K13 is on, which mean the result is (010)2, corresponding to 2)*

**

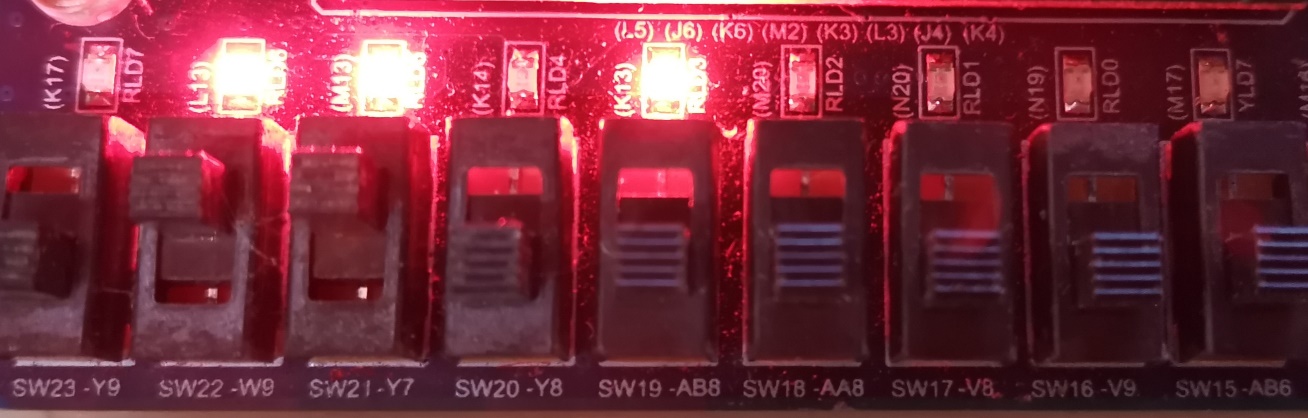
*Step4: set augend(Y7) off(augend=0), and addend be 3(both Y9,W9 on), then the result is below.(K13,M20 are on and K14 is off, which mean the result is (011)2, corresponding to 3)*

**

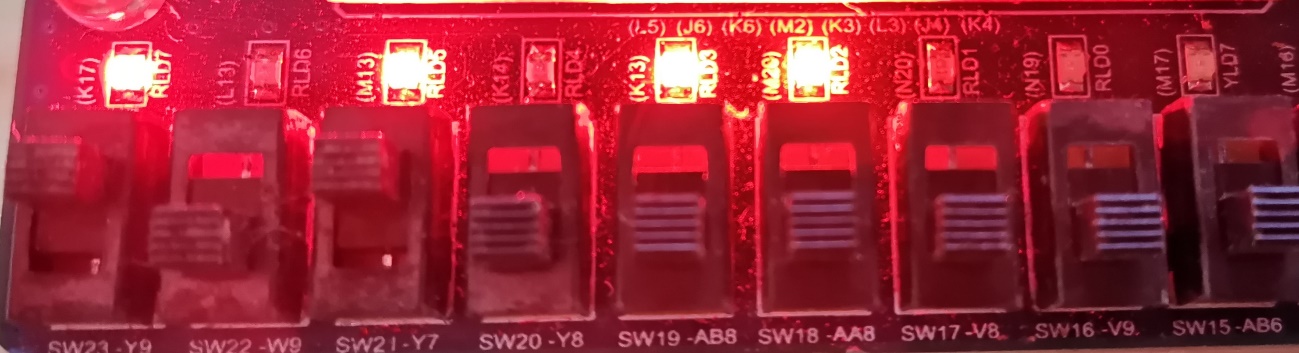
*Step5: set augend(Y7) on(augend=1), and addend be 0(both Y9,W9 off), then the result is below.( K14,K13 are off and M20 is on, which mean the result is (001)2, corresponding to 1)*

**

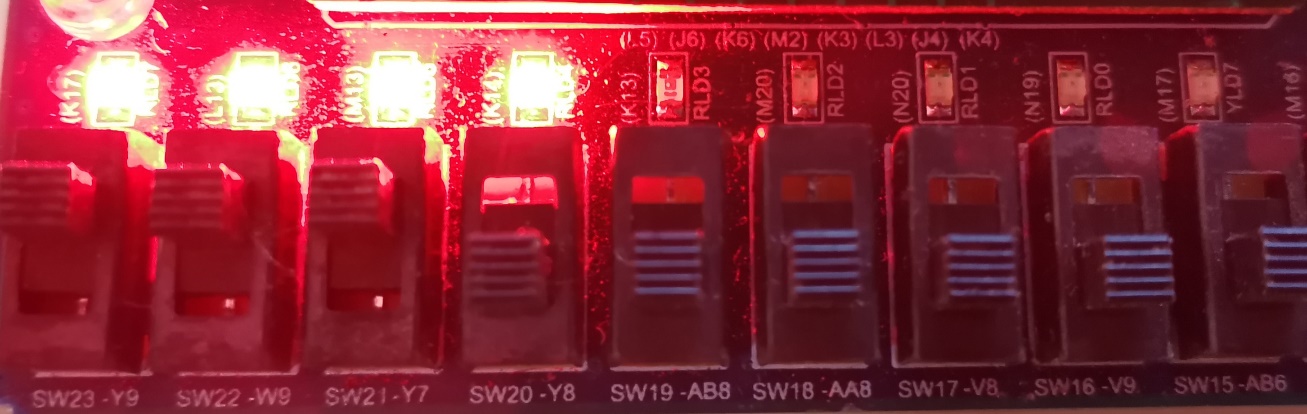
*Step6: set augend(Y7) on(augend=1), and addend be 1(Y9 off,W9 on), then the result is below.(K14,M20 are off and K13 is on, which mean the result is (010)2, corresponding to 2)*

**

*Step7: set augend(Y7) on(augend=1), and addend be 2(Y9 on,W9 off), then the result is below.(K13,M20 are on and K14 is off, which mean the result is (011)2, corresponding to 3)*

**

*Step8: set augend(Y7) on(augend=1), and addend be 3(both Y9,W9 on), then the result is below.(both K13,M20 are off and K14 is on, which mean the result is (100)2, corresponding to 4)*

**

##### the description of operation

*This question is not very difficult, and the operations are very easy. The only difficult I met is that I cannot do simulation the first time. The report told me there were something errors in my simulation code. So I check my simulation code repeatedly, and finally I do simulation successfully.*

*Actually, I really don’t know what should I write in this part (“The description of operation”) because I think the operation is basic. But I am worried about that if I don’t write anything here, my assignment will be lower.*

PART 2: DIGITAL design LAB (Task2)

##### Design

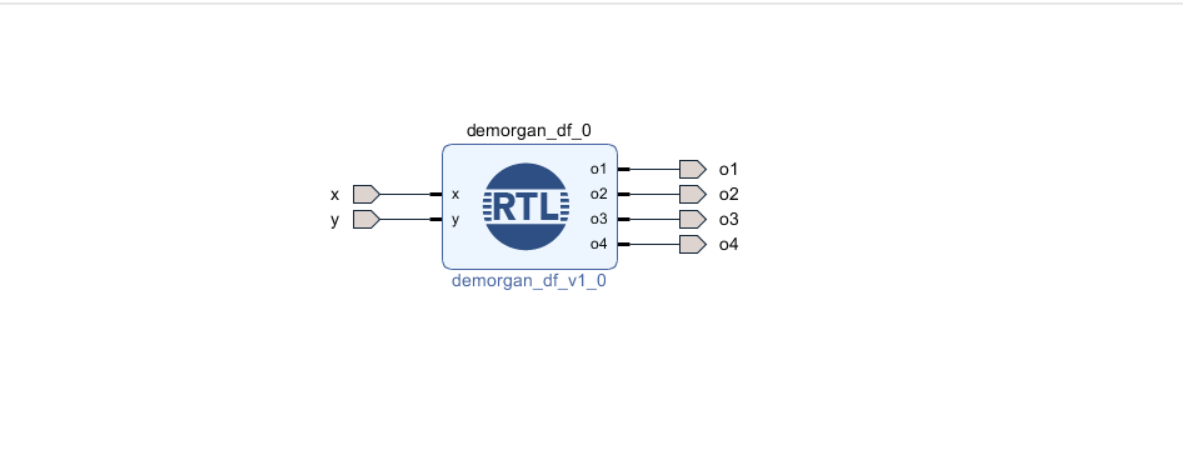
*Verilog Design Code(Data flow style):*

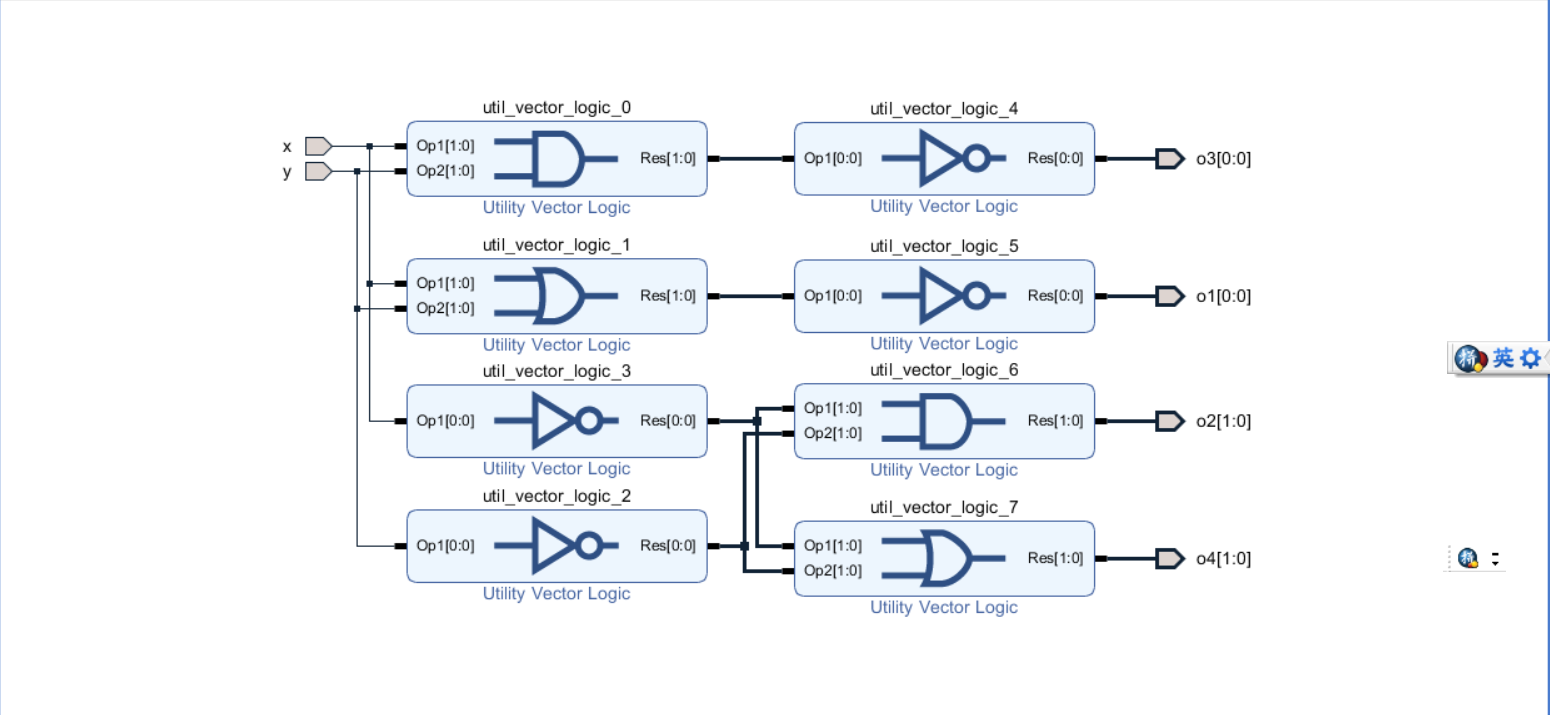
module demorgan\_df(x,y,o1,o2,o3,o4);  
input x,y;  
output o1,o2,o3,o4;  
assign o1 = ~(x|y);  
assign o2 = (~x)&(~y);  
assign o3 = ~(x&y);  
assign o4 = ~x|~y;  
endmodule

*Verilog Design Code(Structured design style):*

module demorgan\_sd(x,y,o1,o2,o3,o4);  
input x,y;  
output o1,o2,o3,o4;  
wire xandy,xory,notx,noty;  
​  
or or1(xory,x,y);  
and and1(xandy,x,y);  
not not1(notx,x);  
not not2(noty,y);  
​  
not not3(o1,xory);  
and and2(o2,notx,noty);  
not not4(o3,xandy);  
or or2(o4,notx,noty);  
endmodule

*Verilog Design Screen Shots(Block design style):*





*Truth-table*

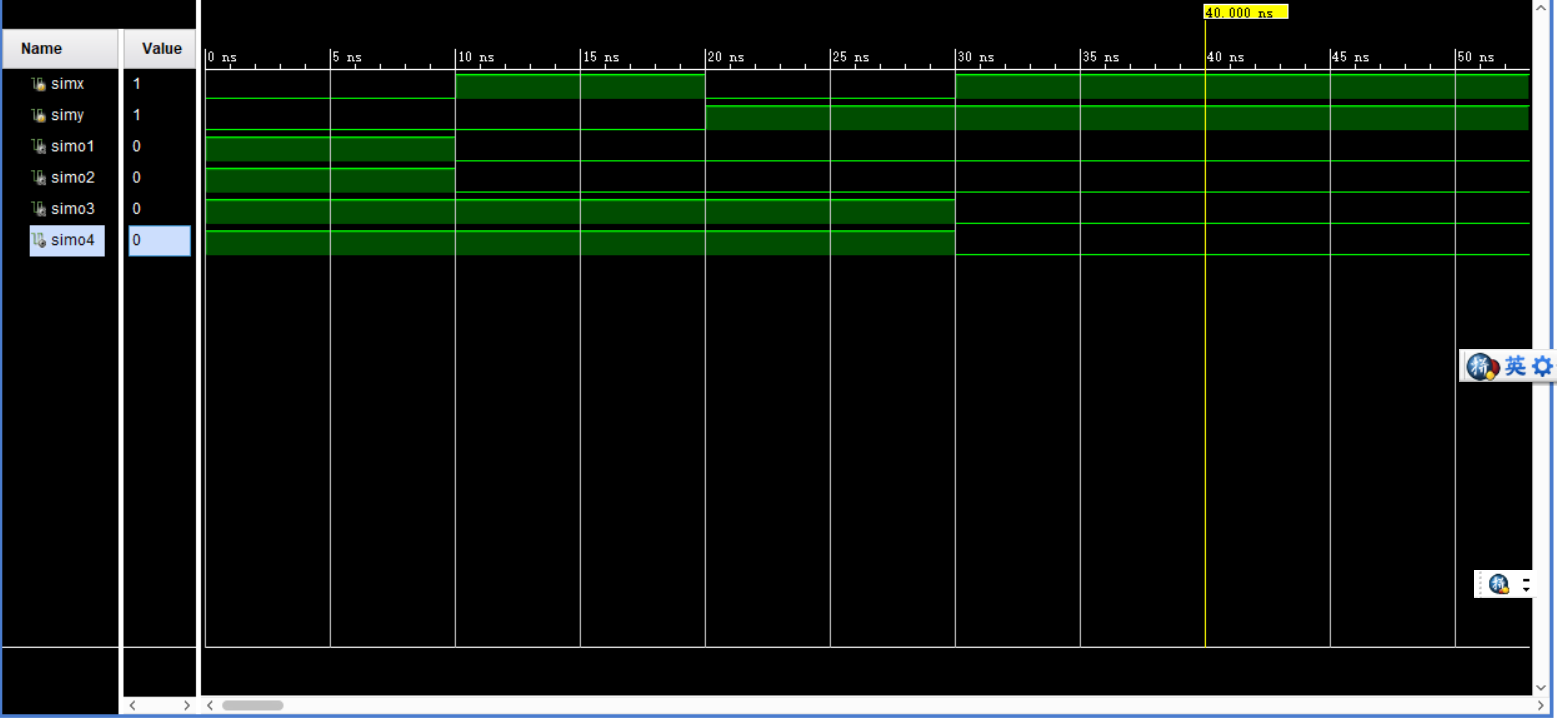
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | (x+y)’ | x’y’ | (xy)’ | x’+y’ |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |

##### simulation

*Verilog Simulation Code:*

module demorgan\_sim( );  
reg simx,simy;  
wire simo1,simo2,simo3,simo4;  
​  
demorgan\_sd u(  
.x(simx),.y(simy),.o1(simo1),.o2(simo2),.o3(simo3),.o4(simo4));  
​  
initial  
begin  
    simx=0;simy=0;  
    #10 simx=1;simy=0;  
    #10 simx=0;simy=1;  
    #10 simx=1;simy=1;  
end  
endmodule

*Wave Form:*



*Analysis:*

*When 0ns – 10ns, simx=0, simy=0, it shows that simo1((x+y)’), simo2(x’y’), simo3((xy)’), simo4(x’+y’) are all equal to 1, which is same as the truth table.*

*When 10ns – 20ns, simx=1, simy=0, it shows that simo1((x+y)’), simo2(x’y’) are equal to 0, simo3((xy)’), simo4(x’+y’) are equal to 1, which is same as the truth table.*

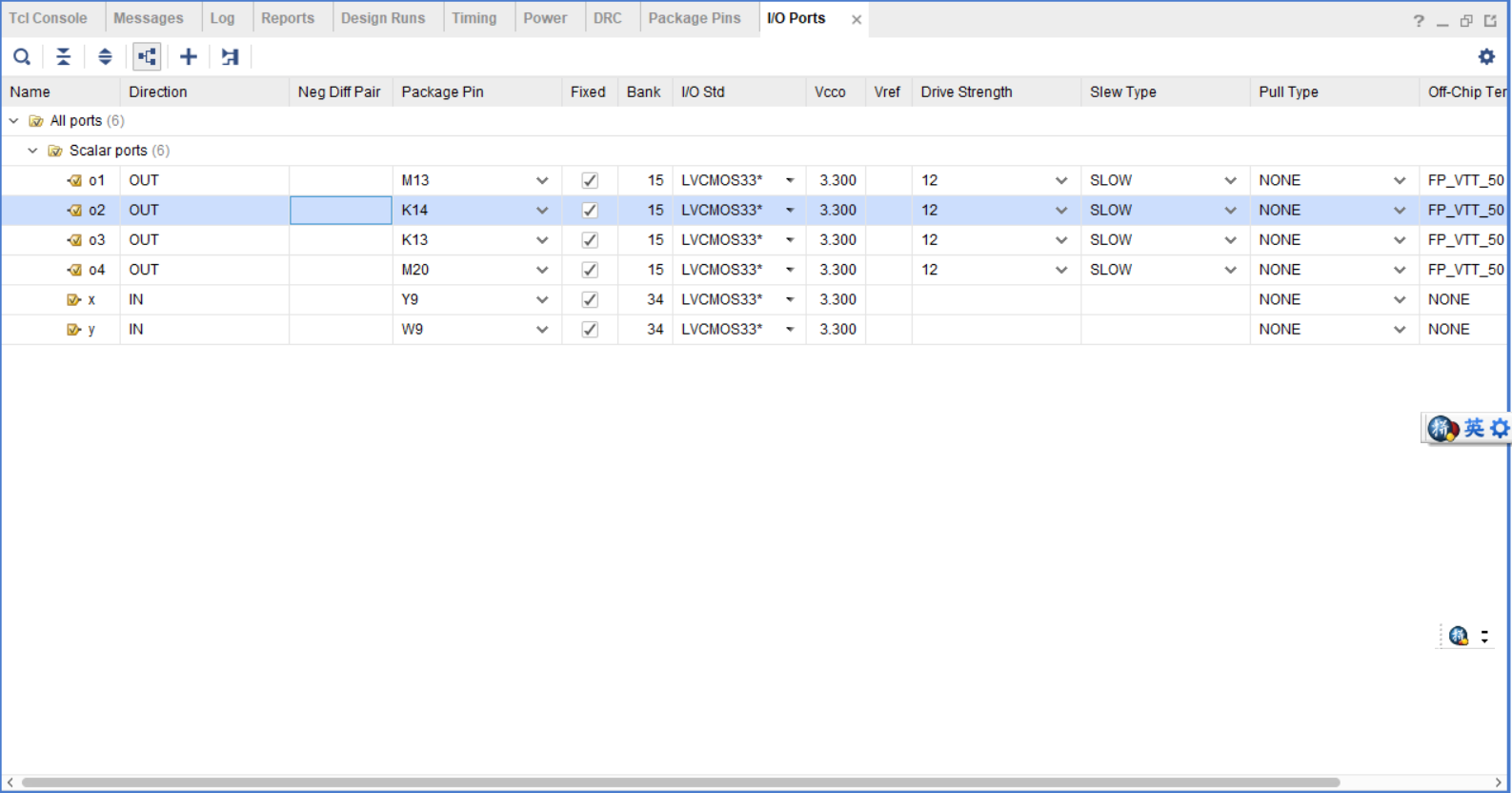
*When 20ns – 30ns, simx=0, simy=1, it shows that simo1((x+y)’), simo2(x’y’) are equal to 0, simo3((xy)’), simo4(x’+y’) are equal to 1, which is same as the truth table.*

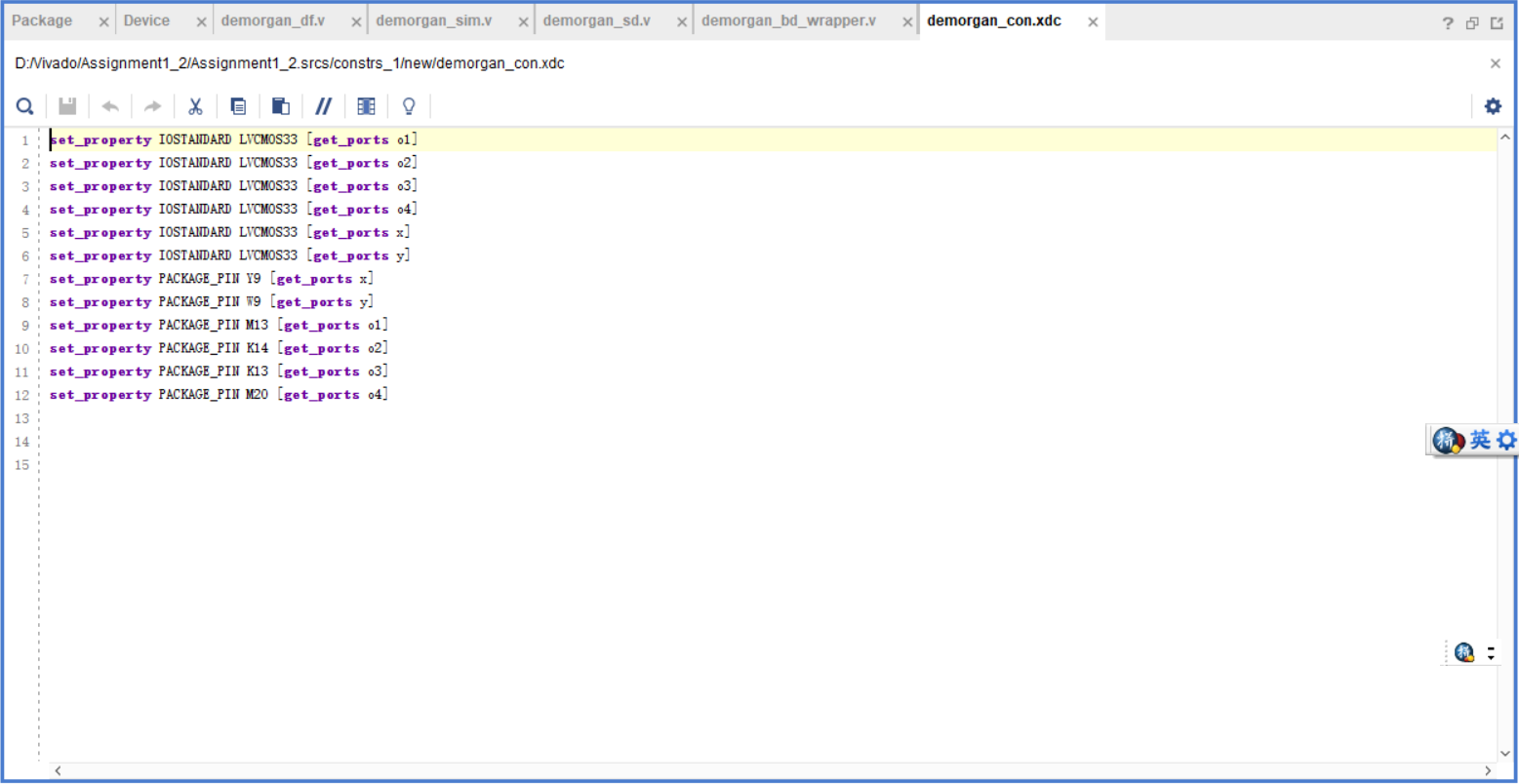
*When 30ns – 40ns, simx=1, simy=1, it shows that simo1((x+y)’), simo2(x’y’), simo3((xy)’), simo4(x’+y’) are all equal to 0, which is same as the truth table.*

*Above all, the simulation result is same as the truth table and the function of design meets our expectation.*

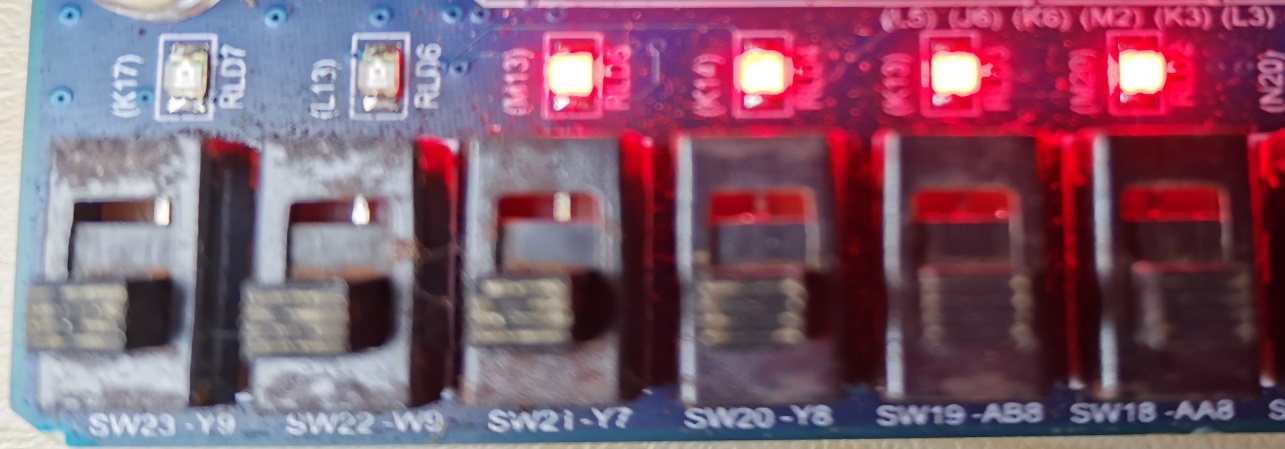
##### Constraint file and the testing

*Constraint File*

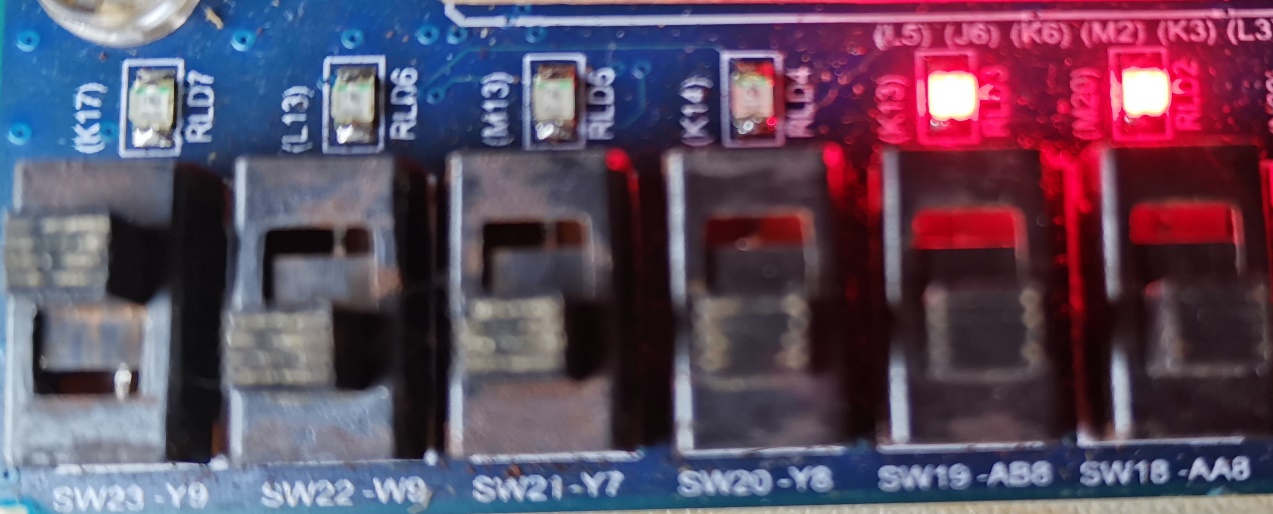




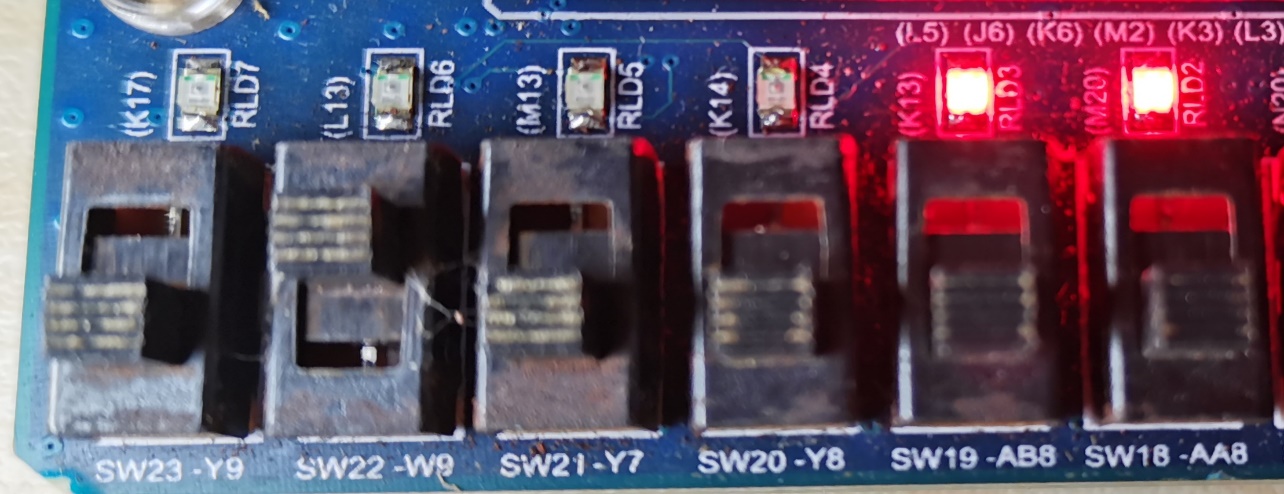
*When both x(Y9) and y(W9) are off, it shows that (x+y)’(M13), x’y’(K14), (xy)’(K13), x’+y’(M20) are all equal to 1.*

**

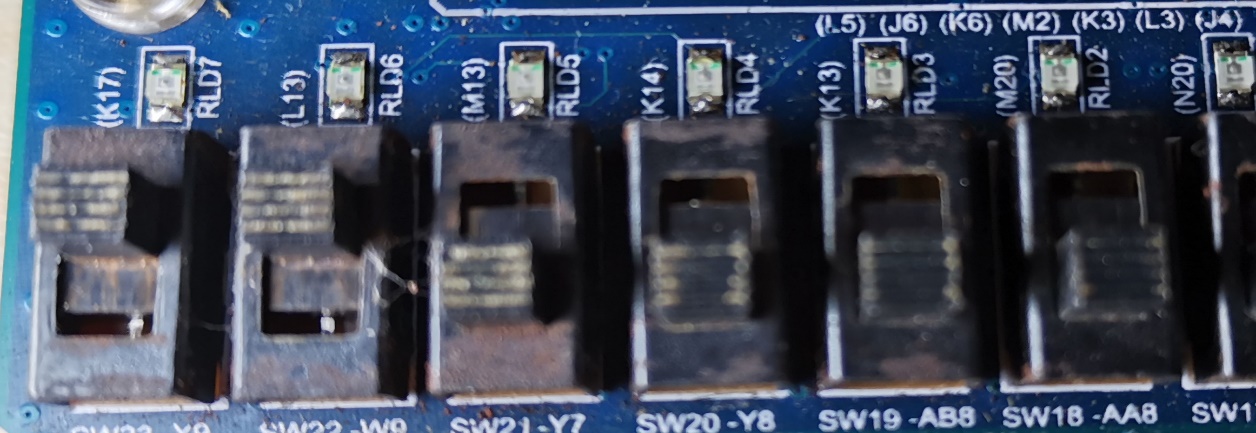
*When x(Y9) is on and y(W9) is off, it shows that (x+y)’(M13), x’y’(K14) are equal to 0, (xy)’(K13), x’+y’(M20) are equal to 1.*

**

*When x(Y9) is off and y(W9) is on, it shows that (x+y)’(M13), x’y’(K14) are equal to 0, (xy)’(K13), x’+y’(M20) are equal to 1.*

**

*When both x(Y9) and y(W9) are on, it shows that (x+y)’(M13), x’y’(K14), (xy)’(K13), x’+y’(M20) are all equal to 0.*

**

##### the description of operation

* *Since I have not used the structured style before, so I spent a lot of time understanding the usage of the structured style.*
* *Since I don’t know whether should I use “add module” or “add IP” in my design file(block design style), I tried both of them.*