

数字逻辑作业

一、开发实现一个四选一的数据选择器；

二、开发实现一个四选一的数据选择器。

1.设计思路：按照老师上课教的流程，创建工程、编写源文件代码、进行综合仿真实现等过程，最后得到正确的四选一数据选择器。

2.设计源文件代码：

```
1. `timescale 1ns / 1ps
2. module MUX41(
3.     input a,
4.     input b,
5.     input c,
6.     input d,
7.     input sel1,
8.     input sel2,
9.     output ot
10. );
11.     reg ot;
12.     always@(a or b or c or d or sel1 or sel2)
13.         case({sel1,sel2})
14.             2'b00:ot=a;
15.             2'b01:ot=b;
16.             2'b10:ot=c;
17.             2'b11:ot=d;
18.         endcase
19. endmodule
```

3.设计仿真测试代码:

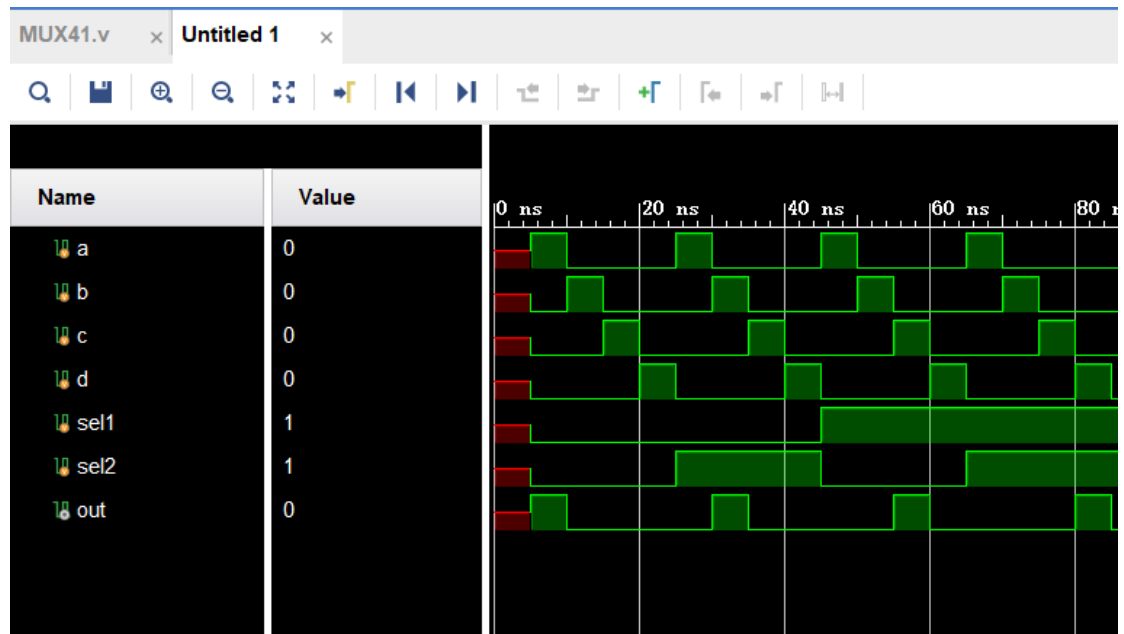
```
1. `timescale 1ns / 1ps
2. module simMUX41(
3.     );
4.     reg a,b,c,d,sel1,sel2;
5.     wire out;
6.     MUX41 testmux41(a,b,c,d,sel1,sel2,out);
7.     initial begin
8.
9.         #5 begin a=1;b=0;c=0;d=0;sel1=0;sel2=0; end
10.        #5 begin a=0;b=1; end
11.        #5 begin b=0;c=1; end
12.        #5 begin c=0;d=1; end
13.
14.        #5 begin a=1;b=0;c=0;d=0;sel1=0;sel2=1; end
15.        #5 begin a=0;b=1; end
16.        #5 begin b=0;c=1; end
17.        #5 begin c=0;d=1; end
18.
19.        #5 begin a=1;b=0;c=0;d=0;sel1=1;sel2=0; end
20.        #5 begin a=0;b=1; end
21.        #5 begin b=0;c=1; end
22.        #5 begin c=0;d=1; end
23.
24.        #5 begin a=1;b=0;c=0;d=0;sel1=1;sel2=1; end
25.        #5 begin a=0;b=1; end
26.        #5 begin b=0;c=1; end
27.        #5 begin c=0;d=1; end
28.        #5 begin a=0;b=0;c=0;d=0; end
```

```

29.     end
30. endmodule

```

4.运行行为仿真，观察信号行为的正确性：



可以发现：

当 sel1=0;sel2=0 时，out 的波形与 a 一致；

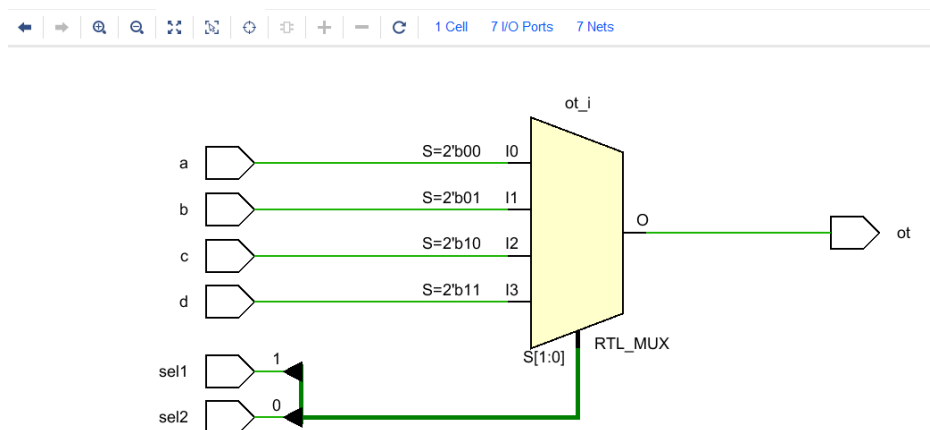
当 sel1=0;sel2=1 时，out 的波形与 b 一致；

当 sel1=1;sel2=0 时，out 的波形与 c 一致；

当 sel1=1;sel2=1 时，out 的波形与 d 一致。

综上所述，源代码设计正确。

5.综合之后，打开 RTL 分析，得到 RTL 级别的电路图：



由电路图也可知，模块设计正确。故四选一数据选择器模块设计正确。仿真模块设计正确，能较好地通过仿真验证模块的正确性。

三、在开发板上实现该模块的功能测试；

1.设计思路：

使用开发板上的 V16 与 V17 两个拨码开关作为信道选择信号的输入，即开关拨向上即为 1，拨向下即为 0，V16 与 V17 分别代表 sel1 和 sel2。使用 L1 作为输出。使用上下左右四个按键分别代表 a,b,c,d 四路输入，T18 代表 a，W19 代表 b，T17 代表 c，U17 代表 d。

这样分配好管脚以后，即可实现四选一数据选择器的功能。通过 V16 V17 两个拨码开关的选择，可以实现信道选择信号的输入，从而选择对应的信号。例如，若 V16 与 V17 的拨码全都在下，也即 sel1=0;sel2=0，此时 out 应该与 a 一致。按下 T18，W19，T17，U17 四个分别代表 a,b,c,d 的按键，发现只有按下代表 a 的 T18 时，LED 灯才会亮。也即此时输出信号与 a 信号一致。说明功能正确。

2.管脚约束

I/O Ports x													
Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
All ports (7)													
Scalar ports (7)													
IN			T18	<input checked="" type="checkbox"/>	14	...	3.300				NONE	NONE	
IN			W19	<input checked="" type="checkbox"/>	14	...	3.300				NONE	NONE	
IN			T17	<input checked="" type="checkbox"/>	14	...	3.300				NONE	NONE	
IN			U17	<input checked="" type="checkbox"/>	14	...	3.300				NONE	NONE	
OUT			L1	<input checked="" type="checkbox"/>	35	...	3.300		12	SLOW	NONE	FP_VTT_50	
IN			V16	<input checked="" type="checkbox"/>	14	...	3.300				NONE	NONE	
IN			V17	<input checked="" type="checkbox"/>	14	...	3.300				NONE	NONE	

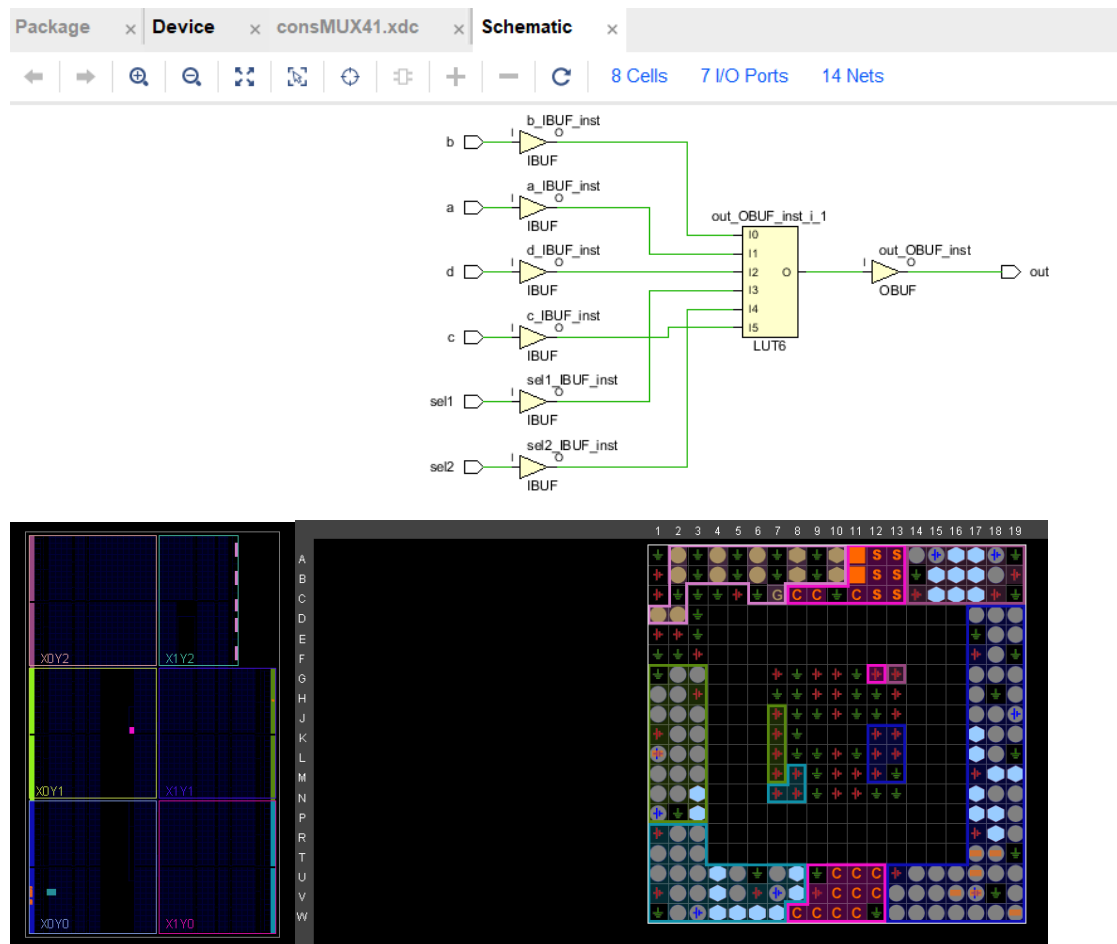
1. set_property PACKAGE_PIN V16 [get_ports sel1]
2. set_property PACKAGE_PIN V17 [get_ports sel2]
3. set_property IOSTANDARD LVCMOS33 [get_ports sel1]
4. set_property IOSTANDARD LVCMOS33 [get_ports sel2]

```

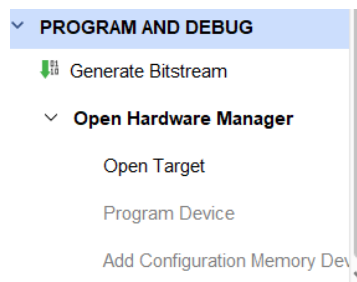
5. set_property PACKAGE_PIN T18 [get_ports a]
6. set_property PACKAGE_PIN W19 [get_ports b]
7. set_property IOSTANDARD LVCMOS33 [get_ports a]
8. set_property IOSTANDARD LVCMOS33 [get_ports b]
9. set_property PACKAGE_PIN T17 [get_ports c]
10. set_property IOSTANDARD LVCMOS33 [get_ports c]
11. set_property PACKAGE_PIN U17 [get_ports d]
12. set_property IOSTANDARD LVCMOS33 [get_ports d]
13. set_property PACKAGE_PIN L1 [get_ports out]
14. set_property IOSTANDARD LVCMOS33 [get_ports out]

```

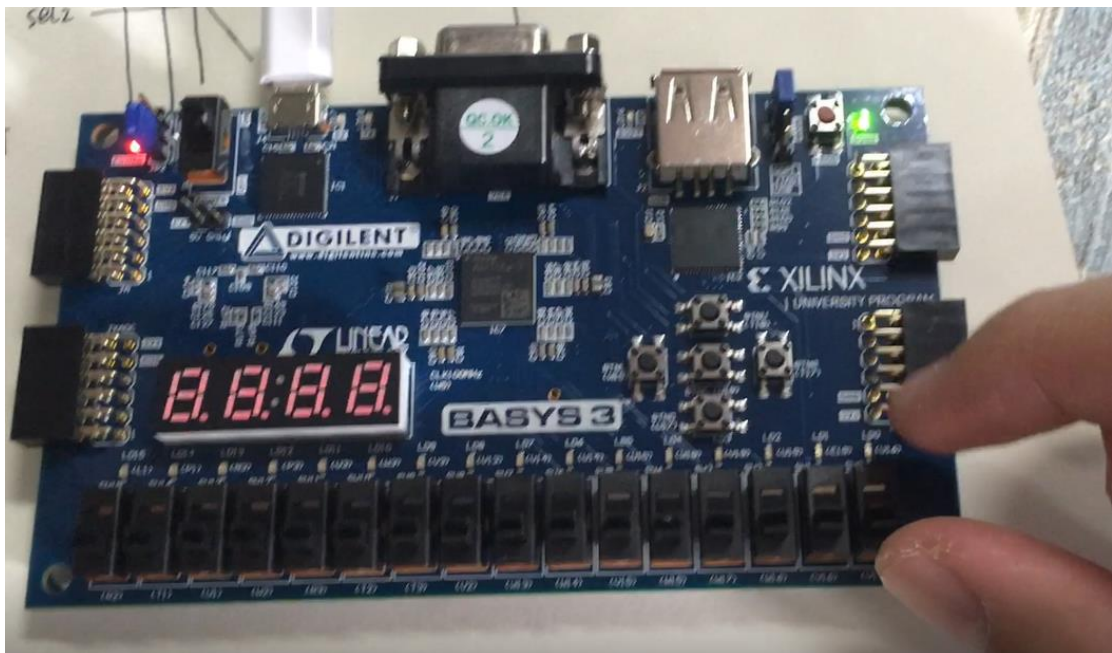
3.综合与实现得到电路图



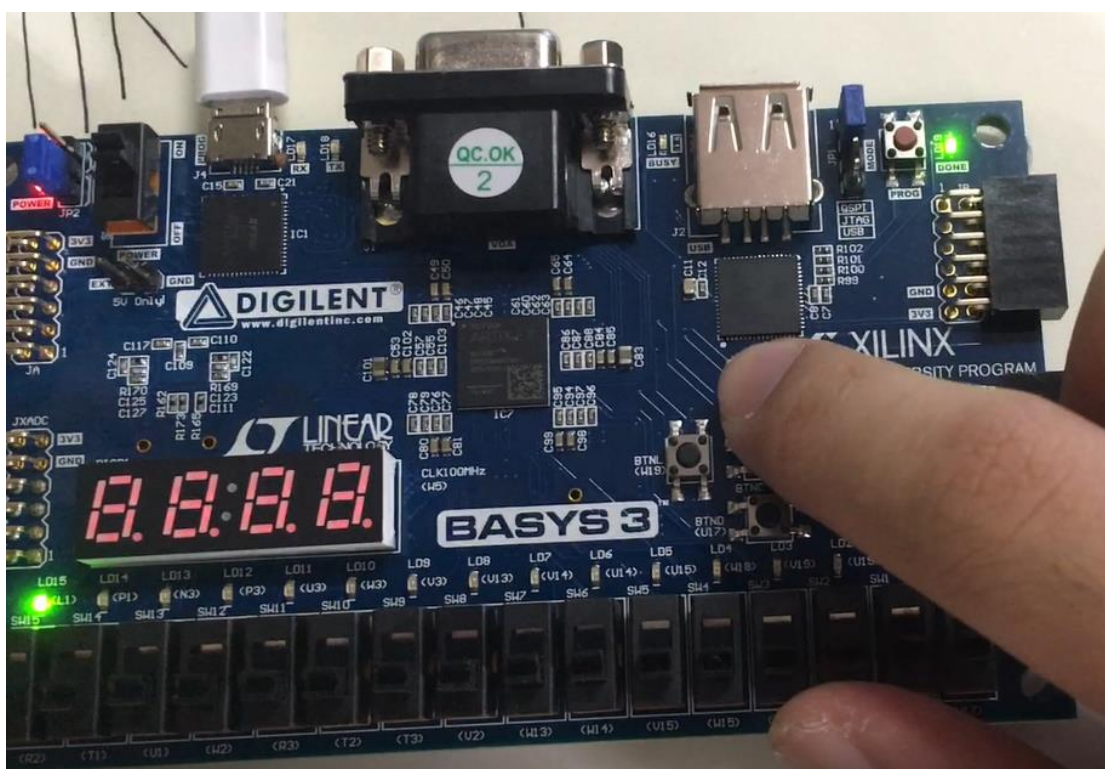
4.生成比特流文件并烧录开发板



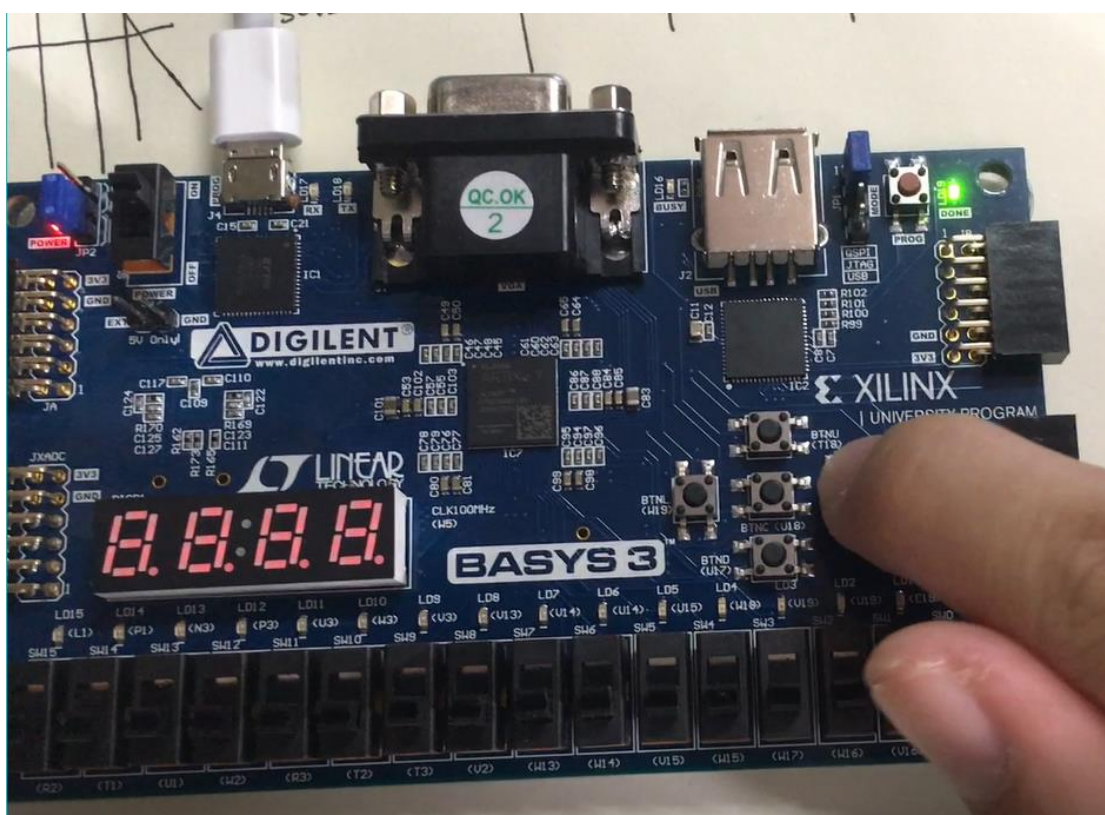
5.在开发板上验证正确性



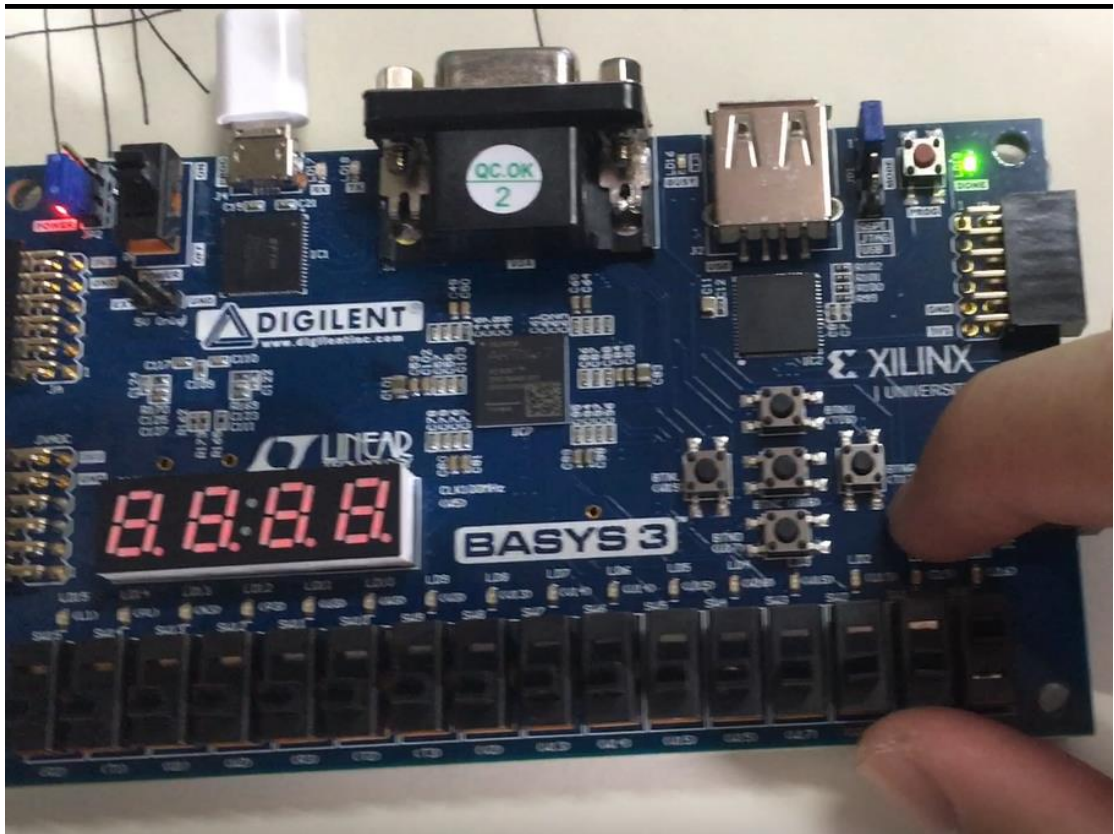
(1) 将信道选择信号置为 0 0



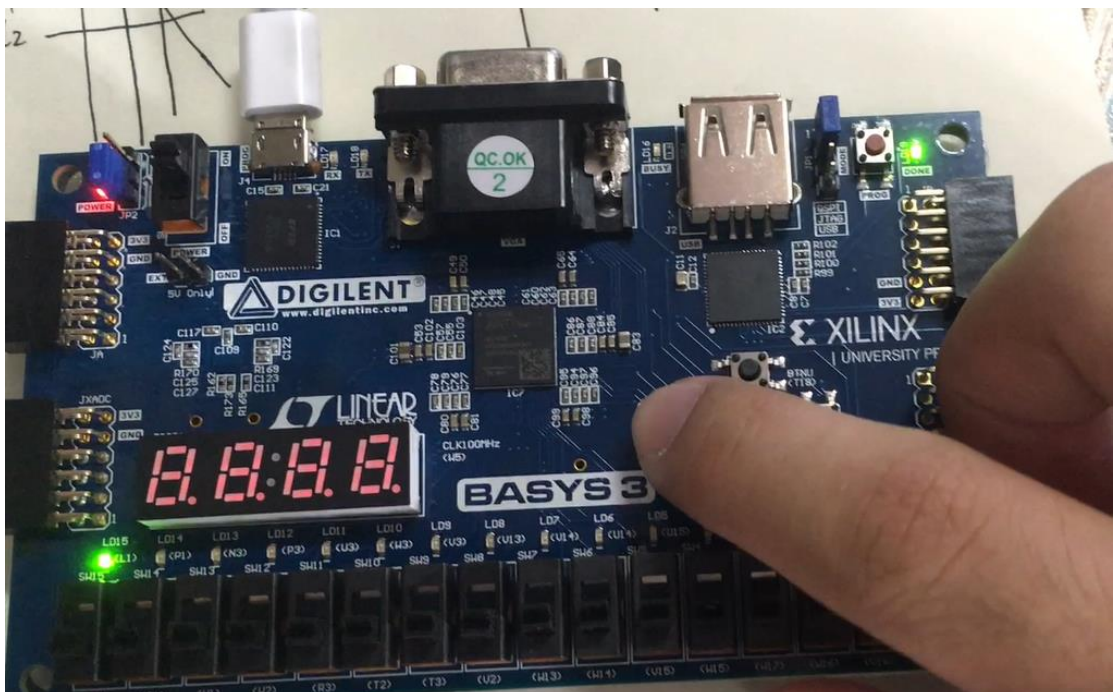
(2) 按下 a 路信号对应的按键，输出信号 LED 灯亮



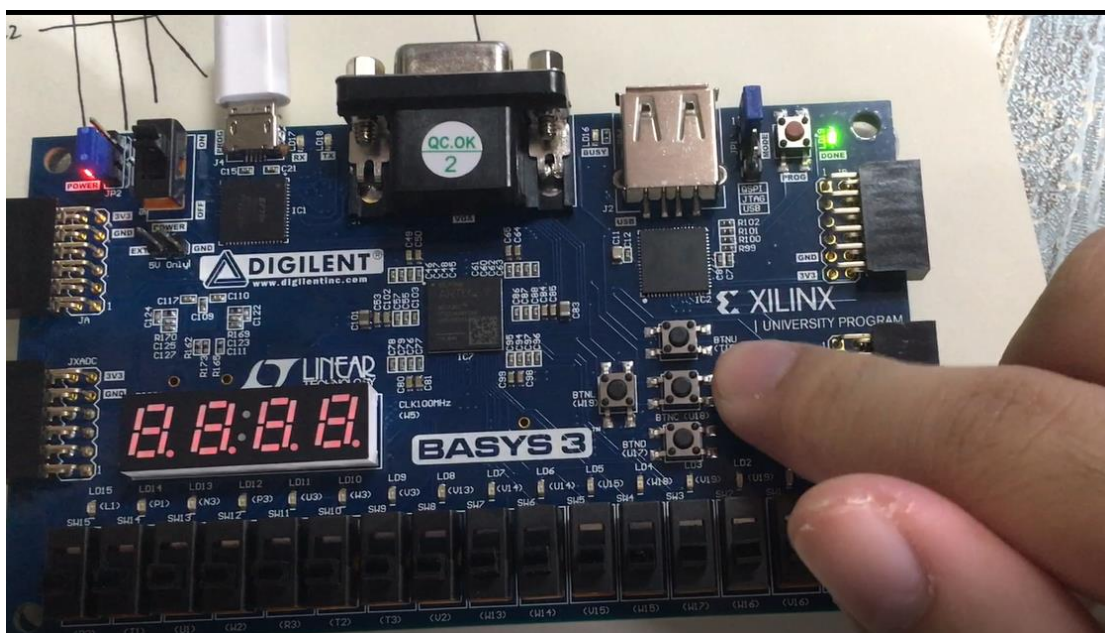
(3) 按下 b,c,d 路信号对应的按键，输出信号 LED 灯均不亮



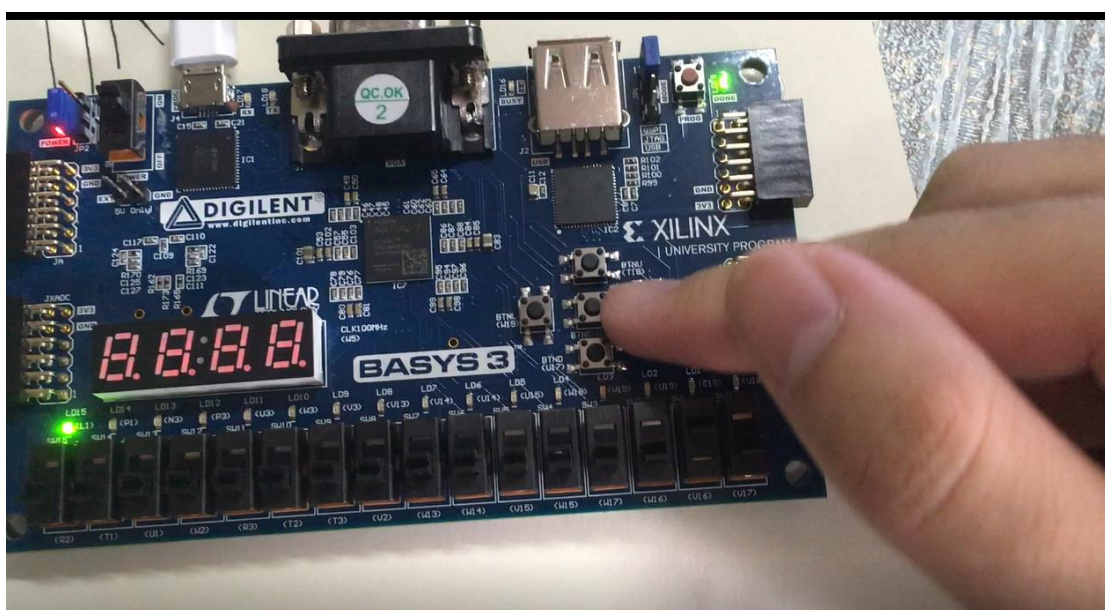
(4) 将信道选择信号置为 0 1



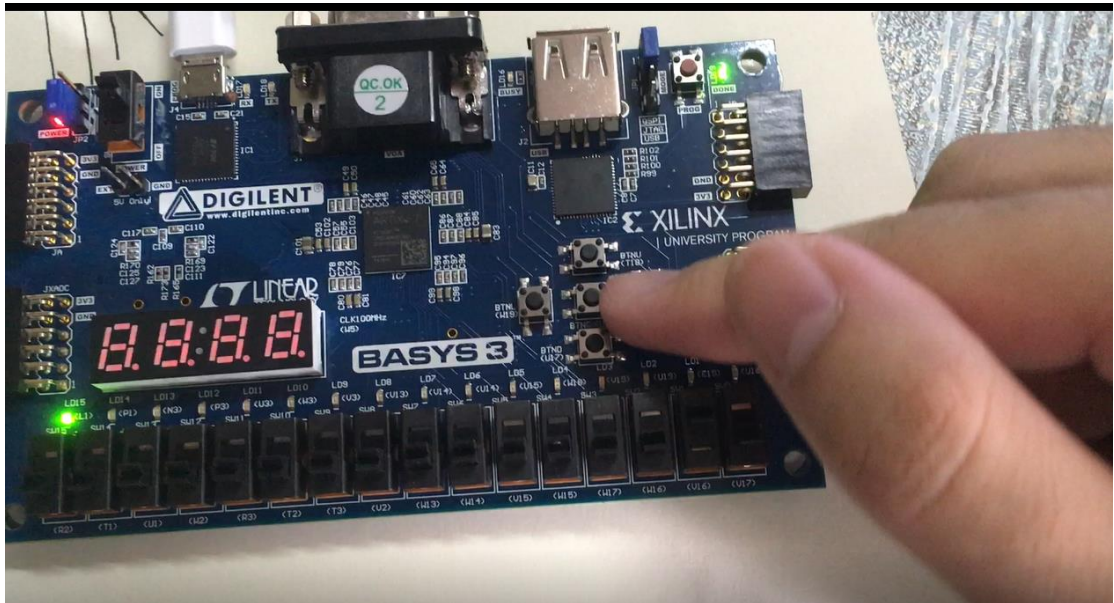
(5) 按下 b 路信号对应的按键，输出信号 LED 灯亮



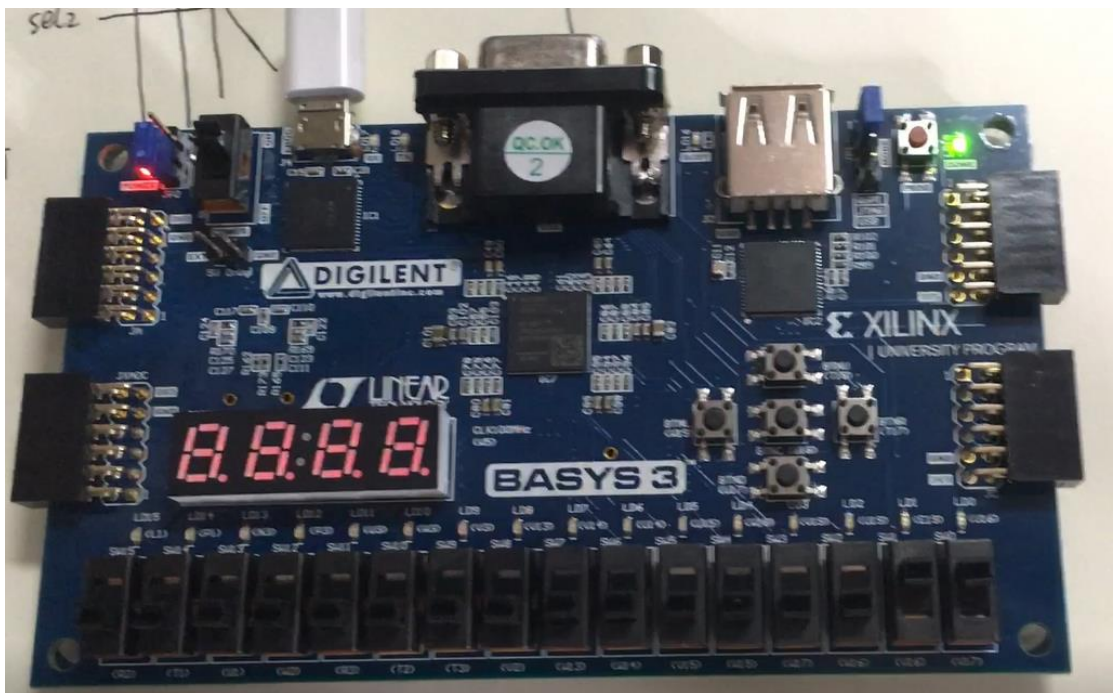
(6) 按下 a,c,d 路信号对应的按键，输出信号 LED 灯均不亮



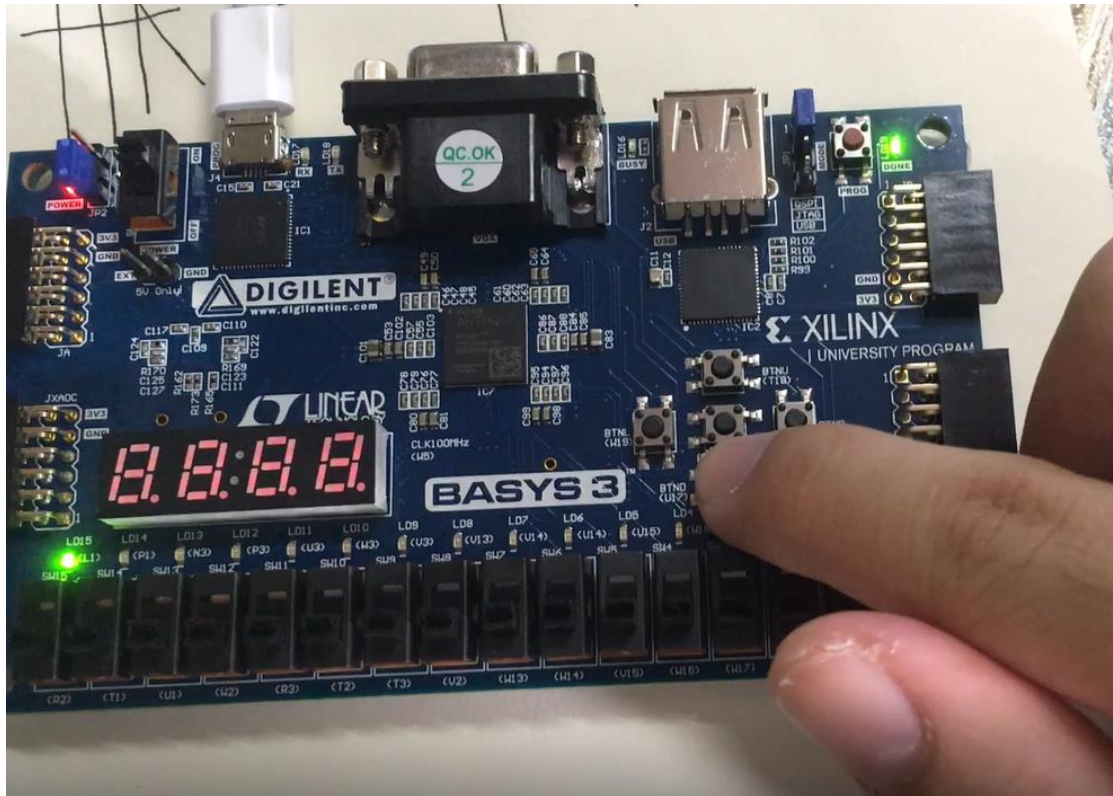
(7) 将信道选择信号置为 1 0



(8) 按下 c 路信号对应的按键，输出信号 LED 灯亮；按下 a,c,d 路信号对应的按键，输出信号 LED 灯均不亮。



(9) 将信道选择信号置为 1 1



(10) 按下 d 路信号对应的按键，输出信号 LED 灯亮；按下 a,b,c 路信号对应的按键，输出信号 LED 灯均不亮。

综上所述：当信道选择信号为 0 0 时，输出信号与输入信号 a 相同；

当信道选择信号为 0 1 时，输出信号与输入信号 b 相同；



当信道选择信号为 1 0 时，输出信号与输入信号 c 相同；

当信道选择信号为 1 1 时，输出信号与输入信号 d 相同；

所以设计的四选一数据选择器功能正确。

四、将该数据选择器模块封装为 IP

按照老师提供的参考资料上提供的封装流程进行封装



Create and Package New IP

This wizard can be used to accomplish following tasks:

- Package a new IP for the Vivado IP Catalog**
This wizard will guide you through the process of creating a new Vivado IP using source files and information from your current project, block design or specified directory.
- Create a new AXI4 Peripheral**
This wizard will guide you through the process of creating a new AXI4 peripheral which includes HDL, driver, software test application, IP Integrator VIP simulation and debug demonstration design.

Click Next to continue


?

< Back

Next >

Finish

Cancel



Create and Package New IP

Package Your Current Project

Select the directory where the IP Definition will be created and the associated options for packaging the current project.

IP location:

Packaging IP in the project

☒ Include .xci files

☐ Include IP generated files

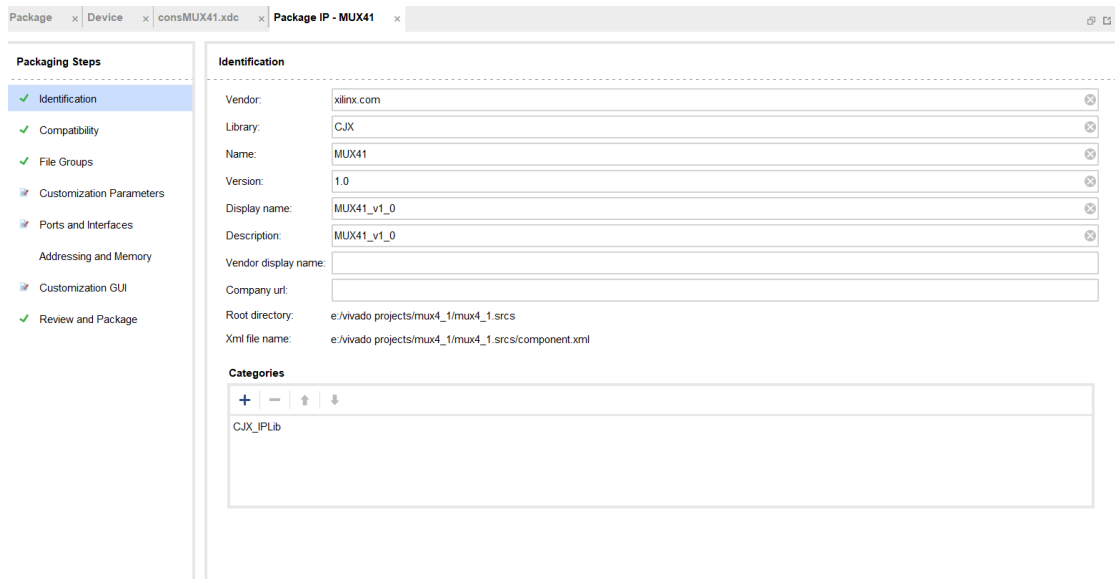
?

< Back

Next >

Finish

Cancel



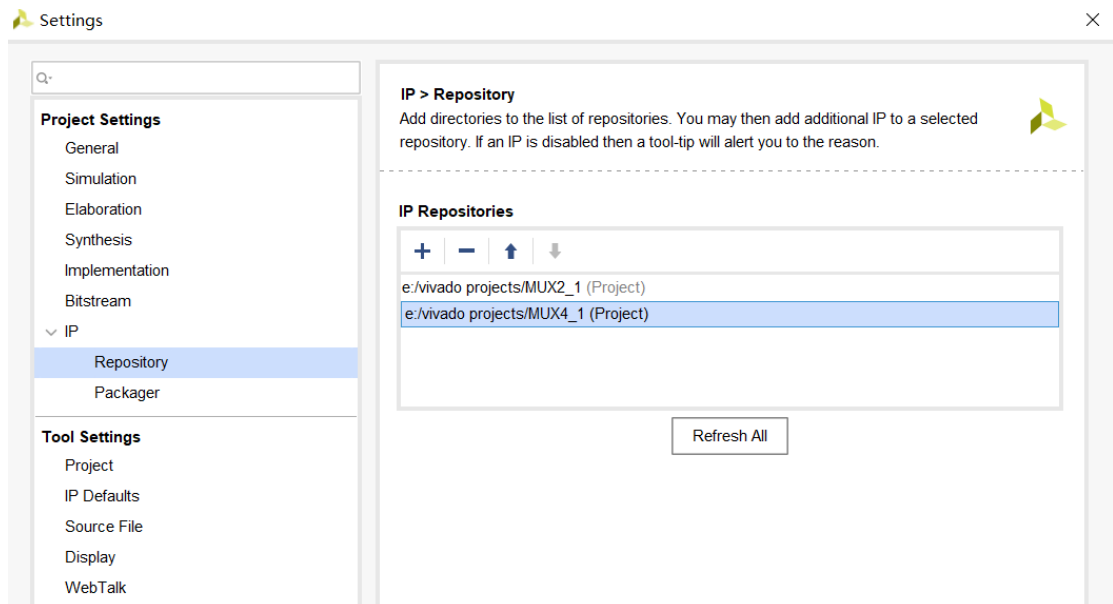
可以看到此模块已经成功封装为了 IP

五、基于上面 IP，实现一个八选一的器件，并仿真测试

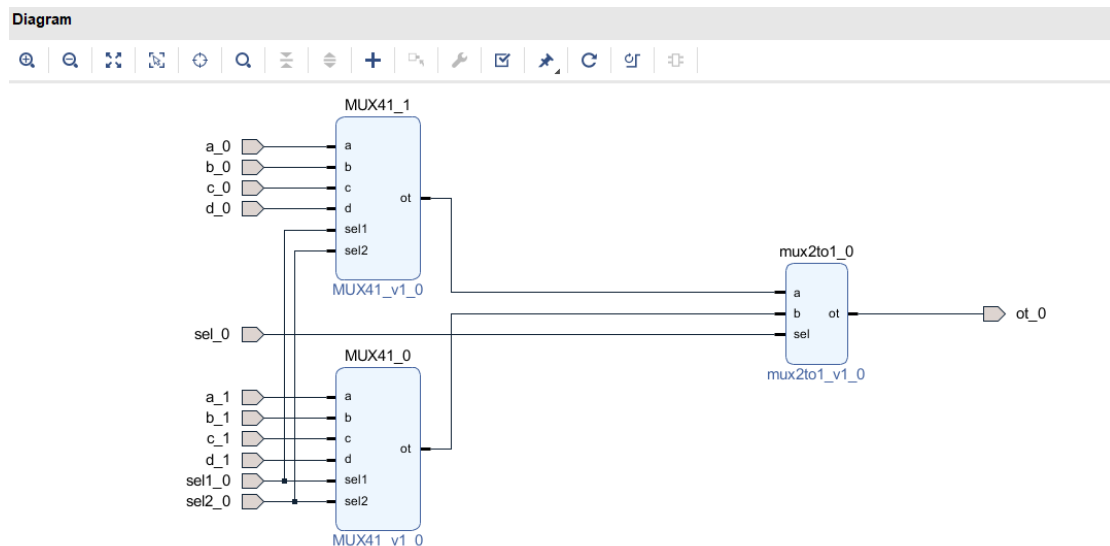
1.前期准备

为实现八选一的功能，另外封装了一个二选一数据选择器。

2.添加 IP

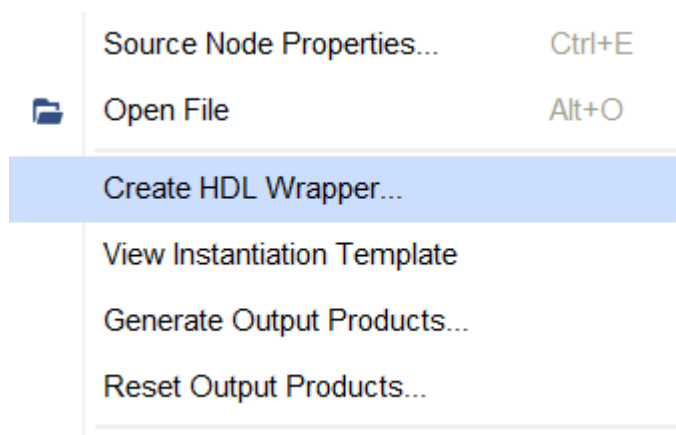


3.将 IP 实例化为器件，再在器件之间连线



连线之后得到如上图所示的图。

4.右键选中 bd 文件，生成源代码



```

1. //Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
2. //-----
3. //Tool Version: Vivado v.2018.3 (win64) Build 2405991 Thu Dec  6 23:38:2
  7 MST 2018
4. //Date       : Tue Oct  1 17:59:44 2019
5. //Host       : LAPTOP-642NED72 running 64-
  bit major release (build 9200)
6. //Command    : generate_target mux81_wrapper.bd
7. //Design     : mux81_wrapper
8. //Purpose    : IP block netlist
9. //-----
10. `timescale 1 ps / 1 ps

```

```
11.
12. module mux81_wrapper
13.     (a_0,
14.      a_1,
15.      b_0,
16.      b_1,
17.      c_0,
18.      c_1,
19.      d_0,
20.      d_1,
21.      ot_0,
22.      sel1_0,
23.      sel2_0,
24.      sel_0);
25.     input a_0;
26.     input a_1;
27.     input b_0;
28.     input b_1;
29.     input c_0;
30.     input c_1;
31.     input d_0;
32.     input d_1;
33.     output ot_0;
34.     input sel1_0;
35.     input sel2_0;
36.     input sel_0;
37.
38.     wire a_0;
39.     wire a_1;
40.     wire b_0;
41.     wire b_1;
42.     wire c_0;
43.     wire c_1;
44.     wire d_0;
45.     wire d_1;
46.     wire ot_0;
47.     wire sel1_0;
48.     wire sel2_0;
49.     wire sel_0;
50.
51.     mux81 mux81_i
52.         (.a_0(a_0),
53.          .a_1(a_1),
54.          .b_0(b_0),
```

```

55.         .b_1(b_1),
56.         .c_0(c_0),
57.         .c_1(c_1),
58.         .d_0(d_0),
59.         .d_1(d_1),
60.         .ot_0(ot_0),
61.         .sel1_0(sel1_0),
62.         .sel2_0(sel2_0),
63.         .sel_0(sel_0));
64. endmodule

```

5.设计编写仿真文件，进行仿真

```

1. `timescale 1ns / 1ps
2. module simMUX81_usingIP(
3. );
4.     reg a_0,a_1,b_0,b_1,c_0,c_1,d_0,d_1,sel1_0,sel2_0,sel_0;
5.     wire ot_0;
6.     mux81_wrapper testmux81(a_0,a_1,b_0,b_1,c_0,c_1,d_0,d_1,ot_0,sel1_0,
7.         sel2_0,sel_0);
8.     initial begin
9.         #5 begin a_0=1;a_1=0;b_0=0;b_1=0;c_0=0;c_1=0;d_0=0;d_1=0;sel1_0=0;se
10.             l2_0=0;sel_0=0; end
11.         #5 begin a_0=0;a_1=1; end
12.         #5 begin a_1=0;b_0=1; end
13.         #5 begin b_0=0;b_1=1; end
14.         #5 begin b_1=0;c_0=1; end
15.         #5 begin c_0=0;c_1=1; end
16.         #5 begin c_1=0;d_0=1; end
17.         #5 begin d_0=0;d_1=1; end
18.
19.         #5 begin a_0=1;a_1=0;b_0=0;b_1=0;c_0=0;c_1=0;d_0=0;d_1=0;sel1_0=0;se
20.             l2_0=0;sel_0=1; end
21.         #5 begin a_0=0;a_1=1; end
22.         #5 begin a_1=0;b_0=1; end
23.         #5 begin b_0=0;b_1=1; end
24.         #5 begin b_1=0;c_0=1; end
25.         #5 begin c_0=0;c_1=1; end
26.         #5 begin c_1=0;d_0=1; end
27.         #5 begin d_0=0;d_1=1; end

```

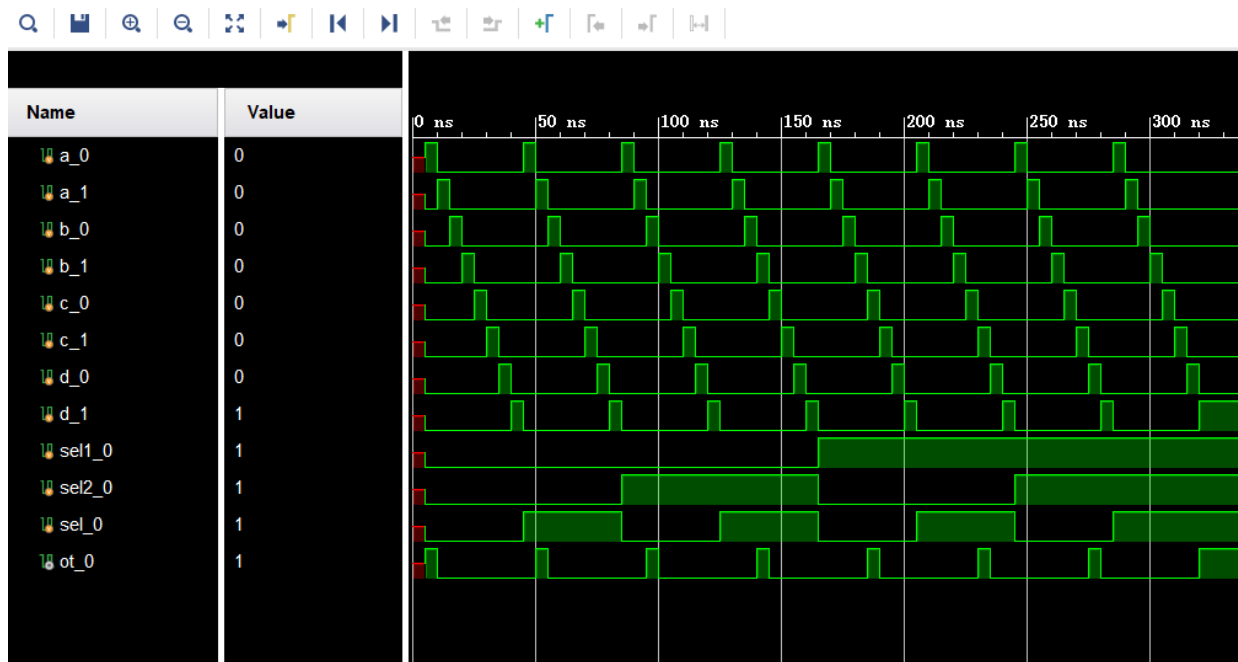
```
28.
29.    #5 begin a_0=1;a_1=0;b_0=0;b_1=0;c_0=0;c_1=0;d_0=0;d_1=0;sel1_0=0;se
    l2_0=1;sel_0=0; end
30.    #5 begin a_0=0;a_1=1; end
31.    #5 begin a_1=0;b_0=1; end
32.    #5 begin b_0=0;b_1=1; end
33.    #5 begin b_1=0;c_0=1; end
34.    #5 begin c_0=0;c_1=1; end
35.    #5 begin c_1=0;d_0=1; end
36.    #5 begin d_0=0;d_1=1; end
37.
38.    #5 begin a_0=1;a_1=0;b_0=0;b_1=0;c_0=0;c_1=0;d_0=0;d_1=0;sel1_0=0;se
    l2_0=1;sel_0=1; end
39.    #5 begin a_0=0;a_1=1; end
40.    #5 begin a_1=0;b_0=1; end
41.    #5 begin b_0=0;b_1=1; end
42.    #5 begin b_1=0;c_0=1; end
43.    #5 begin c_0=0;c_1=1; end
44.    #5 begin c_1=0;d_0=1; end
45.    #5 begin d_0=0;d_1=1; end
46.
47.    #5 begin a_0=1;a_1=0;b_0=0;b_1=0;c_0=0;c_1=0;d_0=0;d_1=0;sel1_0=1;se
    l2_0=0;sel_0=0; end
48.    #5 begin a_0=0;a_1=1; end
49.    #5 begin a_1=0;b_0=1; end
50.    #5 begin b_0=0;b_1=1; end
51.    #5 begin b_1=0;c_0=1; end
52.    #5 begin c_0=0;c_1=1; end
53.    #5 begin c_1=0;d_0=1; end
54.    #5 begin d_0=0;d_1=1; end
55.
56.    #5 begin a_0=1;a_1=0;b_0=0;b_1=0;c_0=0;c_1=0;d_0=0;d_1=0;sel1_0=1;se
    l2_0=0;sel_0=1; end
57.    #5 begin a_0=0;a_1=1; end
58.    #5 begin a_1=0;b_0=1; end
59.    #5 begin b_0=0;b_1=1; end
60.    #5 begin b_1=0;c_0=1; end
61.    #5 begin c_0=0;c_1=1; end
62.    #5 begin c_1=0;d_0=1; end
63.    #5 begin d_0=0;d_1=1; end
64.
65.    #5 begin a_0=1;a_1=0;b_0=0;b_1=0;c_0=0;c_1=0;d_0=0;d_1=0;sel1_0=1;se
    l2_0=1;sel_0=0; end
66.    #5 begin a_0=0;a_1=1; end
```

```

67.     #5 begin a_1=0;b_0=1; end
68.     #5 begin b_0=0;b_1=1; end
69.     #5 begin b_1=0;c_0=1; end
70.     #5 begin c_0=0;c_1=1; end
71.     #5 begin c_1=0;d_0=1; end
72.     #5 begin d_0=0;d_1=1; end
73.
74.     #5 begin a_0=1;a_1=0;b_0=0;b_1=0;c_0=0;c_1=0;d_0=0;d_1=0;sel1_0=1;se
        l2_0=1;sel_0=1; end
75.     #5 begin a_0=0;a_1=1; end
76.     #5 begin a_1=0;b_0=1; end
77.     #5 begin b_0=0;b_1=1; end
78.     #5 begin b_1=0;c_0=1; end
79.     #5 begin c_0=0;c_1=1; end
80.     #5 begin c_1=0;d_0=1; end
81.     #5 begin d_0=0;d_1=1; end
82.
83.     end
84. endmodule

```

6.运行行为仿真，观察信号行为 correctness:



可以发现：

当 sel1=0;sel2=0;sel=0 时，out 的波形与 a0 一致；

当 sel1=0;sel2=0;sel=1 时，out 的波形与 a1 一致；

当 $sel1=0;sel2=1;sel=0$ 时, out 的波形与 b_0 一致;

当 $sel1=0;sel2=1;sel=1$ 时, out 的波形与 b_1 一致;

当 $sel1=1;sel2=0;sel=0$ 时, out 的波形与 c_0 一致;

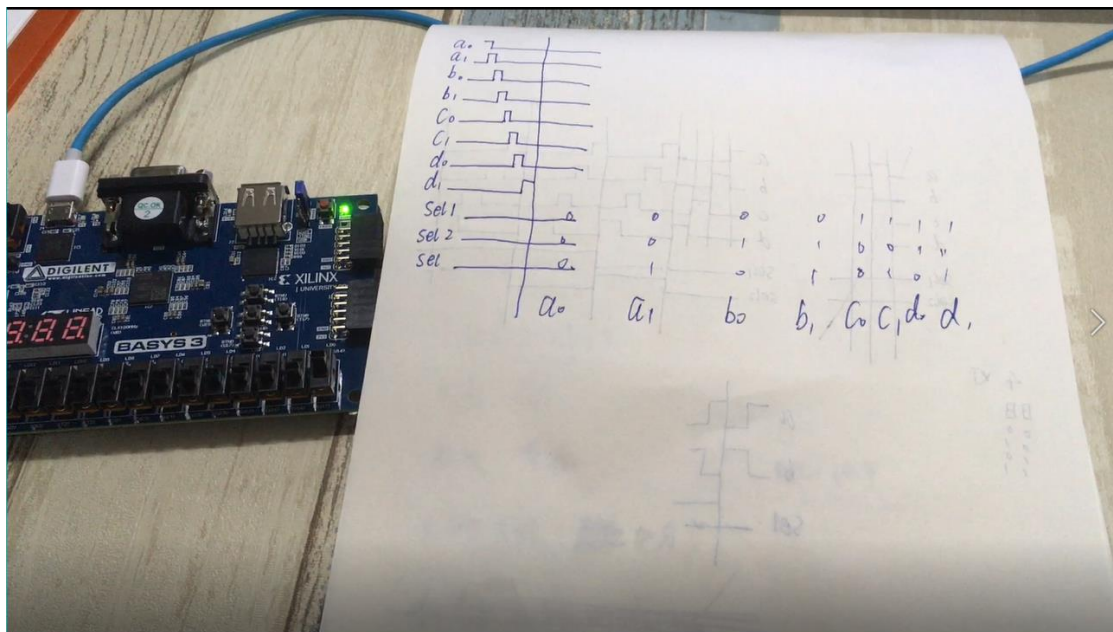
当 $sel1=1;sel2=0;sel=1$ 时, out 的波形与 c_1 一致;

当 $sel1=1;sel2=1;sel=0$ 时, out 的波形与 d_0 一致;

当 $sel1=1;sel2=1;sel=1$ 时, out 的波形与 d_1 一致;

通过仿真结果, 综上所述, 源代码设计正确。

7. 烧录进开发板进行功能验证 (额外做的)



思路: 因为开发板上的按钮不足 8 个, 于是使用左边 8 个拨码开关作为输入信号, 最右边两个拨码开关作为信道选择信号, LED 灯 L1 作为输出信号。其他思路与四选一时完全一致。具体验证详见附件中的视频。