《数字逻辑》实验报告

姓名		**年级**	2022 级
学号		**专业、班级**	计算机科学与技术（卓越） 1 班

实验名称	实验三 4 选 1 多路选择器		
实验时间	2023.10.12	**实验地点**	计算机机房 DS1410
实验成绩		**实验性质**	**□验证性 ☑ 设计性 □综合性**

教师评价：

□算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；

□实验结果正确； □语法、语义正确； □报告规范；

评语：

评价教师签名（电子签名）：

一、实验目的

1.熟悉 Vivado 开发环境，掌握设计流程，包括如何用 Verilog 语言设计电路、仿真、综合、实现与下载，并学会 IP 核的封装与调用。

2.学会调用 IP 核的方法和通过 Block Design 设计电路的方法。

3.通过实验，学会多路选择器的功能，以及使用 Block Design 和 Verilog HDL 语言设计多路选择器。

4.熟悉仿真文件的写法，并熟悉将程序写入开发板的流程。

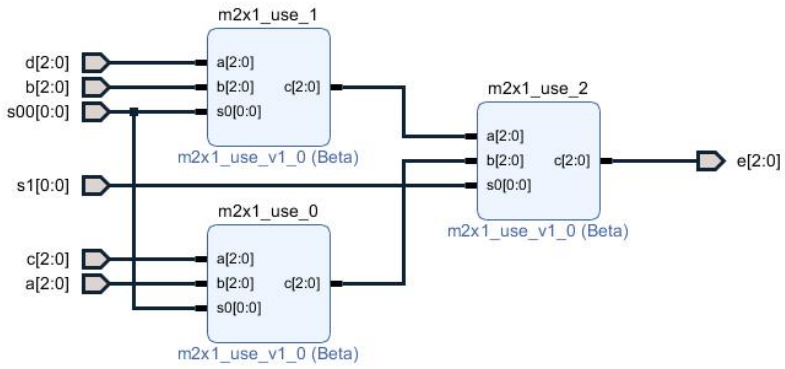
二、实验项目内容

设计一个三位 4 选 1 多路选择器电路，分别用 Block Design 和 verilog HDL 语言编写代码两种方式实现，通过仿真、看 RTL 电路图、下载到板子验证其正确性，并比较不同方法实现的异同（逻辑资源、RTL 电路图、仿真波形等方面）。

三、实验设计

1.方法一:Block Design 的方法：

首先设计 2 选 1 的选择器，并将其封装成 IP 核，并在 4 选 1 选择器设计中调用 2 选 1 选择器的 IP 核来进行实现。



（1）实验原理：先通过次两次二选一选择器选出四者其二，最后在使用一次二选一选择器成功实现四选一。

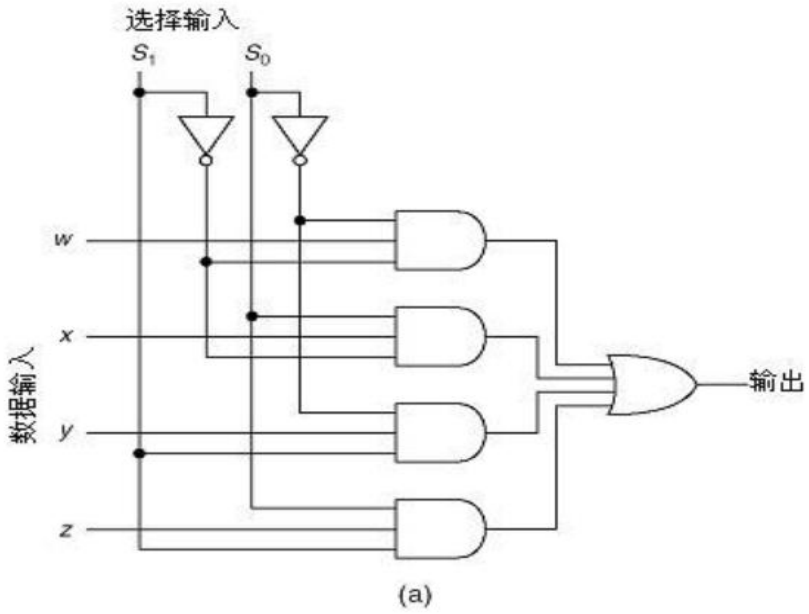
（2）原理公式：

$$f = S_1(\mathrm{d}s_{00} + b\overline{S_{00}}) + \overline{S_1}(cs_{00} + a\overline{s_{00}})$$

（3）其真值表：

S1	S00	f
0	0	a
0	1	b
1	0	c
1	1	d

2.verilog HDL 语言编写方法：



(a)

（1）实验原理：

①输入信号由两个非门来控制输入情况；

②输出情况由四个与门和三个非门进行综合来控制；

③除了 s0 和 s1 以为外的所有信号均为 3 位。

（2）原理公式：

$$f = \overline{S_1}\overline{s_0}w + \overline{S_1}s_0x + s_1\overline{S_0}y + S_1S_0z$$

（3）逻辑真值表：

考虑到 s0 与是 s1 和其他数据位宽不同，所以代码中利用 if else 语句来进行类似模拟。模拟后真值表如图：

S1	S0	f
0	0	w
0	1	x
1	0	y
1	1	z

四、实验过程或算法

1.Block design 语言编写：

（1）首先设计 2 选一选择器 IP 核：

①设计代码如下：

代码注释：编写 ab 两个宽度为 WIDTH 的输入数据，1 个宽度为 1 的 s0 输入数据，一个 c 宽度为 WIDTH 输出数据。用 if 语句进行判断，当 S0=1 时，c=a；当 s0=0 时，c=b；

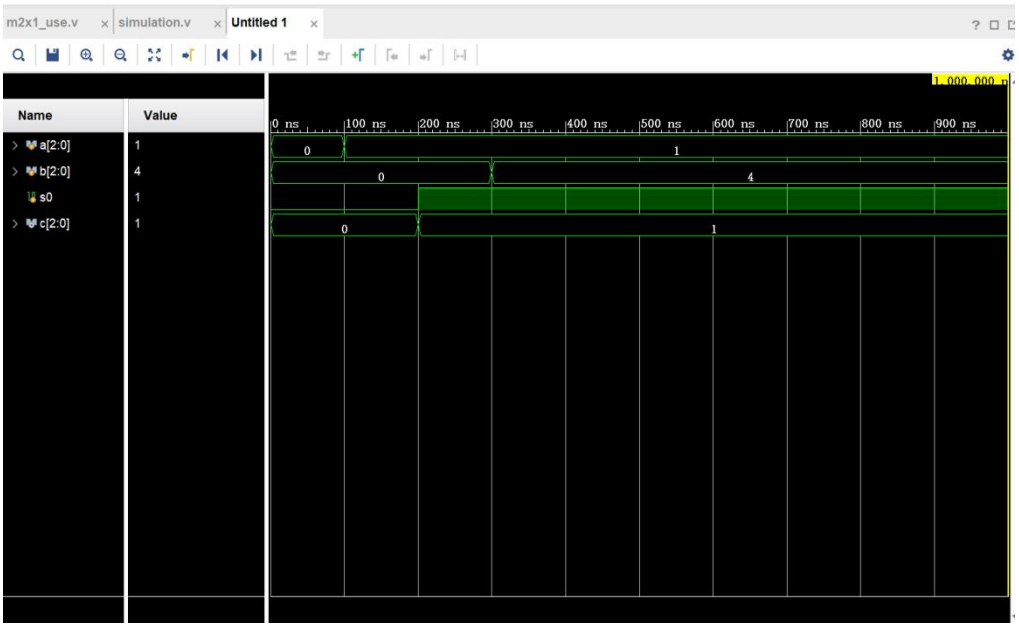
```
module m2x1_use
#(parameter WIDTH=3)
(
input [(WIDTH-1):0] a,
input [(WIDTH-1):0] b,
input [0:0] s0,
output reg [(WIDTH-1):0] c

    );
    always @(a, b, c, s0)
    begin
    if(s0==1)
    c=a;
    else if(s0==0)
    c=b;
    end
endmodule
```

②进行仿真，仿真代码及结果如下：

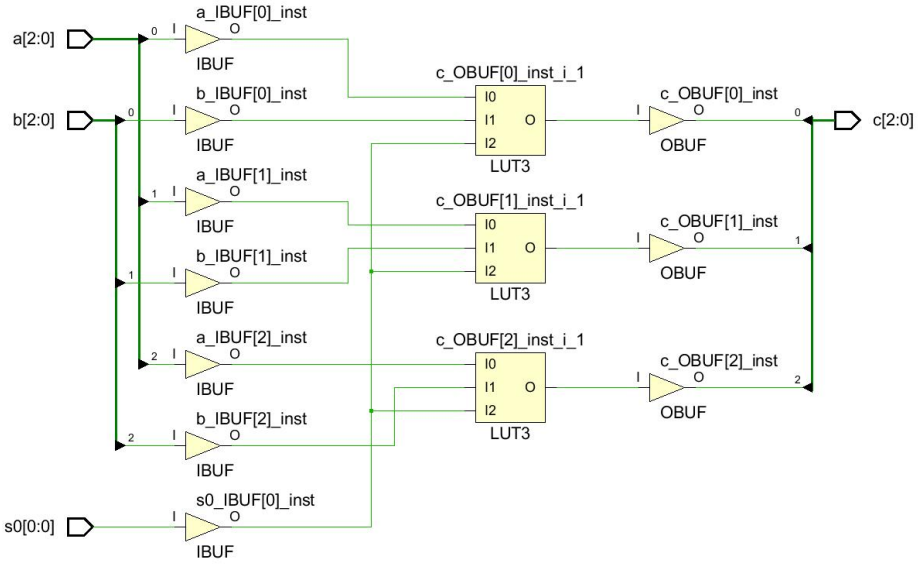
代码注释：首先按照设计文件设出相应需要的数据，然后进行实例

化，最后每隔 100 个时间单位对某些数据进行更改从而达到仿真效果。

```
module simulation(

    );
    reg [2:0] a=3'b000;
    reg [2:0] b=3'b000;
    reg s0=0;
    wire [2:0] c;
    m2x1_use #(3) u(a,b,s0,c);
    initial begin
    #100 a=3'b001;
    #100 s0=1;
    #100 b=3'b100;
    end

endmodule
```



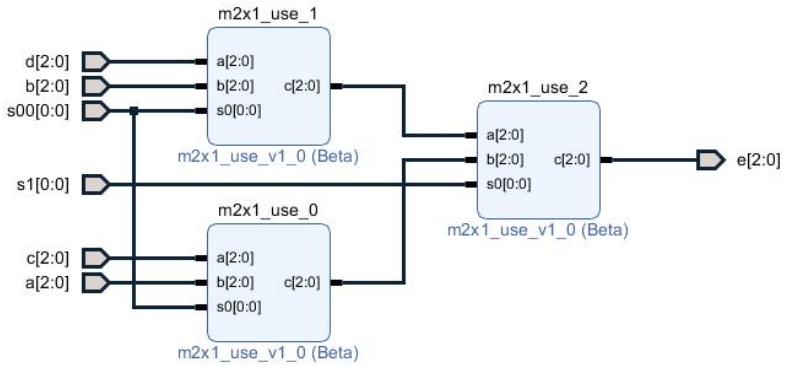
③进行 RTL 分析：



④确认无误后实现、管脚分配、下载程序至开发板观察运行情况，并包装成 IP 核。

（2）IP 核的调用与 block design 设计

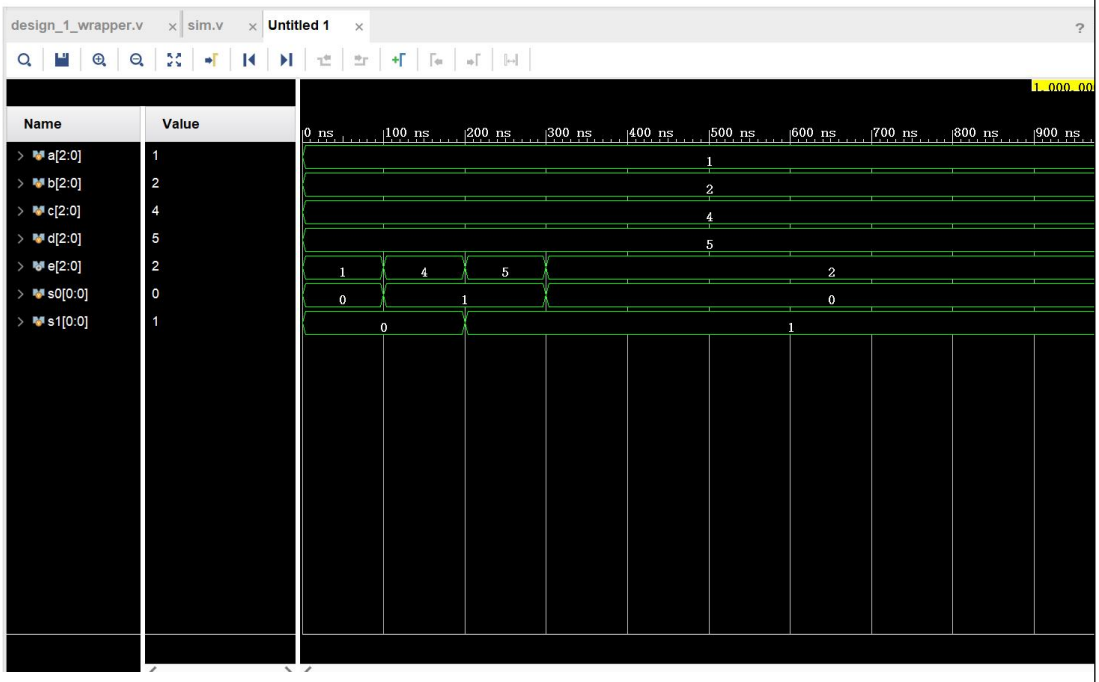
①创建 4 选 1 工程项目，建立 block design 项目，导入并调用 2 选 1IP 核，按照实验原理进行连接，并创建 HDL wapper,生成一个完整的电路设计。



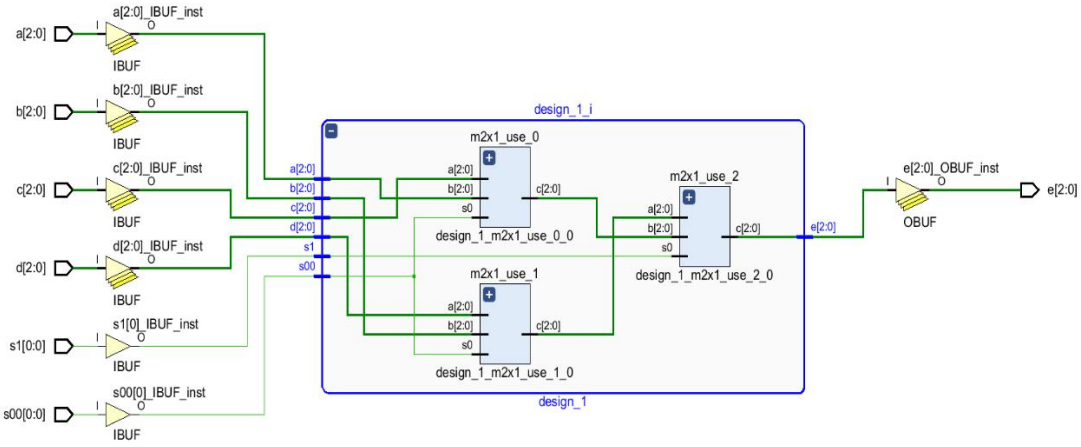
②进行仿真验证并查看仿真图：

代码注释：根据电路设计上的数据要求设置相应数据，然后进行实例化，最后每隔 100 个时间单位对某些数据进行更改从而达到仿真效果

```
module sim(

    );
  reg [2:0]a=3'b001;
  reg [2:0]b=3'b010;
  reg [2:0]c=3'b100;
  reg [2:0]d=3'b101;
  wire [2:0]e;
  reg [0:0]s0;
  reg [0:0]s1;
  design_1 u (.a(a),.b(b),.c(c),.d(d),.e(e),.s00(s0),.s1(s1));
  initial begin
  s0=0;
  s1=0;
  #100 s0=1;
  #100 s1=1;
  #100 s0=0;
  end

endmodule
```



③进行 RTL 分析：



④先综合，然后进行实现、管脚分配：

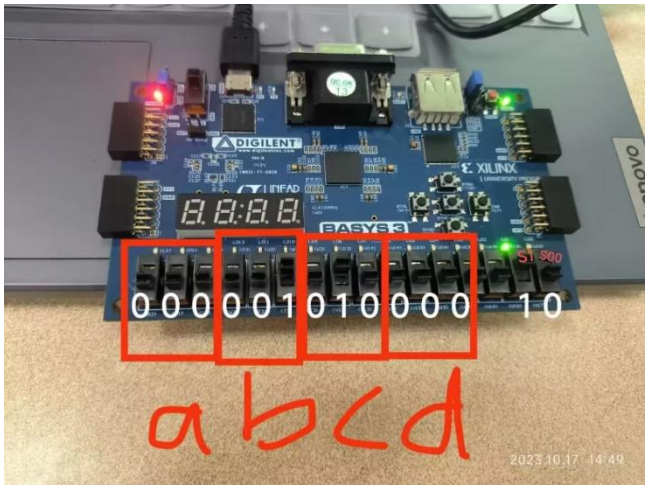
Package × Device × design_1_wrapper.v × sim.v × Scher

D:/VIVADO/m4x12x1/m4x12x1.srcs/constrs_1/new/cons.xdc

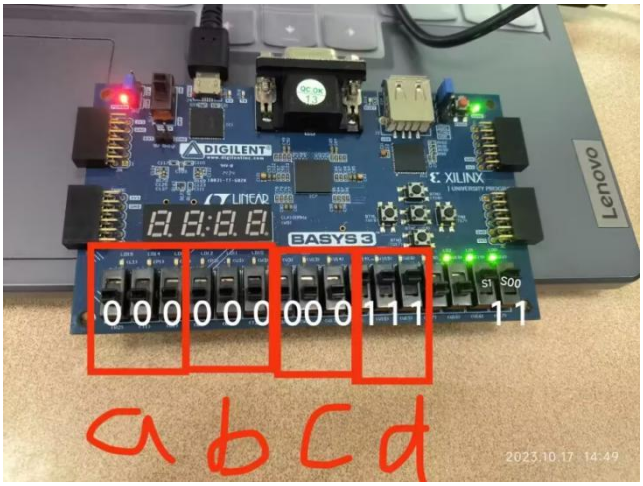
```
set_property IOSTANDARD LVCMOS18 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {b[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {c[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {c[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {c[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {d[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {d[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {d[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {e[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {e[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {e[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {s1[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {s00[0]}]
set_property PACKAGE_PIN R2 [get_ports {a[2]}]
set_property PACKAGE_PIN T1 [get_ports {a[1]}]
set_property PACKAGE_PIN U1 [get_ports {a[0]}]
set_property PACKAGE_PIN T3 [get_ports {c[2]}]
set_property PACKAGE_PIN V2 [get_ports {c[1]}]
set_property PACKAGE_PIN W13 [get_ports {c[0]}]
set_property PACKAGE_PIN W14 [get_ports {d[2]}]
set_property PACKAGE_PIN V15 [get_ports {d[1]}]
set_property PACKAGE_PIN W15 [get_ports {d[0]}]
set_property PACKAGE_PIN U19 [get_ports {e[2]}]
set_property PACKAGE_PIN E19 [get_ports {e[1]}]
set_property PACKAGE_PIN U16 [get_ports {e[0]}]

set_property PACKAGE_PIN V17 [get_ports {s1[0]}]
set_property PACKAGE_PIN V16 [get_ports {s00[0]}]
set_property PACKAGE_PIN W2 [get_ports {b[2]}]
set_property PACKAGE_PIN R3 [get_ports {b[1]}]
set_property PACKAGE_PIN T2 [get_ports {b[0]}]
```

⑤生成比特流，打开硬件程序并打开硬件目标，在开发板上验证自己的

逻辑程序。



a=2’b000,b=2’b001,c=2’b010,d=2’b000,s00=0,s1=1 时， f=b=2’b010



a=2’ b000,b=2’ b000,c=2’ b000,d=2’ b111,s00=1,s1=1 时， f=d=2’ b111

a=2’ b101,b=2’ b000,c=2’ b000,d=2’ b000,s00=0,s1=0 时， f=a=2’ b101

当 a=2’b000,b=2’b101,c=2’b000,d=2’b000,s00=1,s1=0 时，f=d=2’b111.

经过以上检验，与真值表和仿真图所相符。

2.（方法二）Verilog 语言程序设计：

（1） 创建设计文件，并使用 if -else 语句进行实现

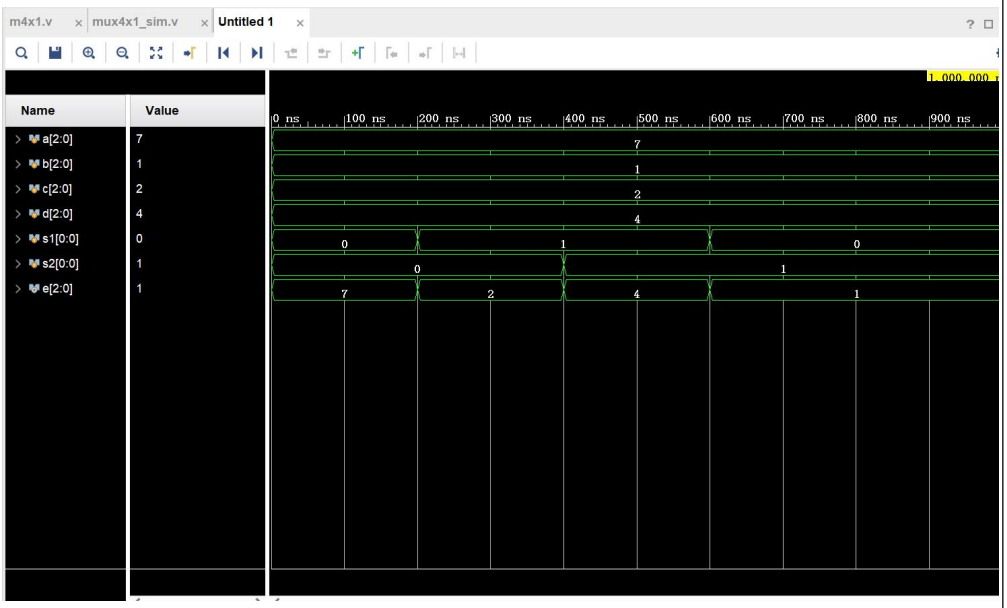
代码注释：设置 abcd 四个宽度为 3 的输入数据，s1，s2 两个宽度为 1 的输入数据和 e 这个宽度为 1 的输出数据，采用 if 语句来决定当 s1,s2 取相应值时，e 等于 abcd 当中那一个的值。

```
module m4x1(
input [2:0] a,
input [2:0] b,
input [2:0] c,
input [2:0] d,
input [0:0] s1,
input [0:0] s2,
output reg [2:0]e
    );
    always @(a, b, c, d, s1, s2)
    begin
    if(s1==0&&s2==0)
    e=a;
     else if(s1==0&&s2==1)
    e=b;
    else if(s1==1&&s2==0)
    e=c;
    else if(s1==1&&s2==1)
    e=d;

  end
endmodule
```

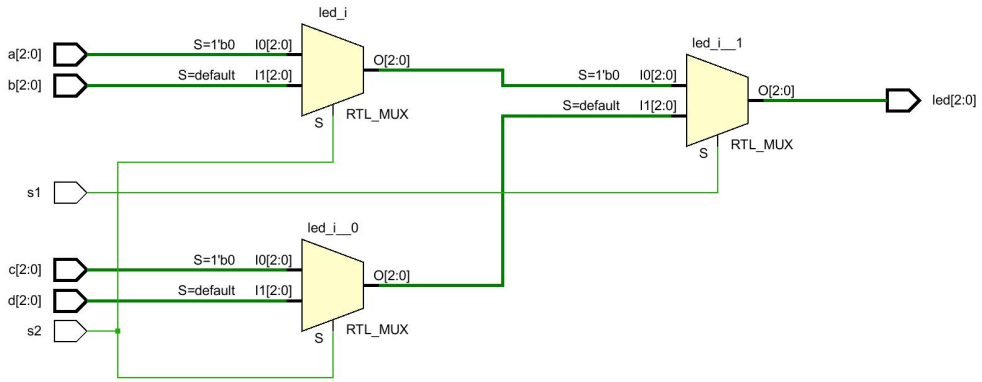
（2） 创建仿真文件并进行仿真，观察仿真图：

代码注释：根据设计文件中数据要求进行设置数据并进行实例化，最后每隔 100 个时间单位对某些数据进行更改从而达到仿真效果

```
module mux4x1_sim(

    );
        reg [2:0] a=3'b111;
    reg [2:0] b=3'b001;
    reg [2:0] c=3'b010;
    reg [2:0] d=3'b100;
    reg [0:0]s1=0;
    reg [0:0] s2=0;
    wire [2:0] e;
    m4x1 u (.a(a),.b(b),.c(c),.d(d),.s1(s1),.s2(s2),.e(e));
    initial
    begin
    #200 s1=1;
    #200 s2=1;
    #200 s1=0;
    end
endmodule
```



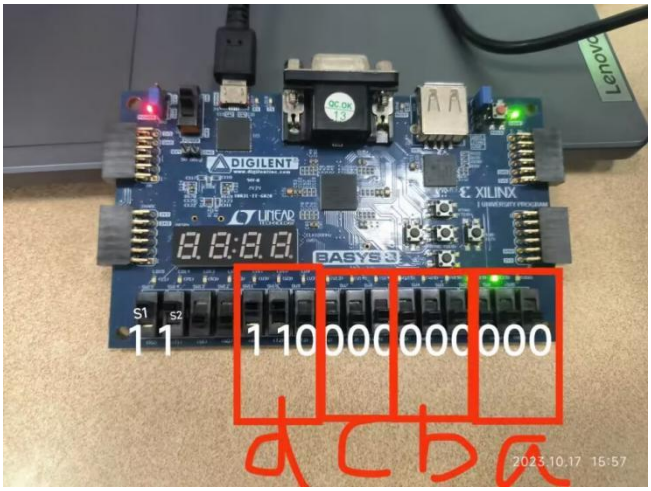
（3） 进行 RTL 分析：



（4） 综合、实现、并进行管脚分配：

```
set_property PACKAGE_PIN U19 [get_ports {e[2]}]
set_property PACKAGE_PIN E19 [get_ports {e[1]}]
set_property PACKAGE_PIN W16 [get_ports {a[2]}]
set_property PACKAGE_PIN V16 [get_ports {a[1]}]
set_property PACKAGE_PIN V17 [get_ports {a[0]}]
set_property PACKAGE_PIN U16 [get_ports {e[0]}]
set_property PACKAGE_PIN V15 [get_ports {b[2]}]
set_property PACKAGE_PIN W15 [get_ports {b[1]}]
set_property PACKAGE_PIN W17 [get_ports {b[0]}]
set_property PACKAGE_PIN V2 [get_ports {c[2]}]
set_property PACKAGE_PIN W13 [get_ports {c[1]}]
set_property PACKAGE_PIN R3 [get_ports {d[2]}]
set_property PACKAGE_PIN T2 [get_ports {d[1]}]
set_property PACKAGE_PIN T3 [get_ports {d[0]}]
set_property PACKAGE_PIN W14 [get_ports {c[0]}]
set_property PACKAGE_PIN R2 [get_ports {s1[0]}]
set_property PACKAGE_PIN T1 [get_ports {s2[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {b[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {c[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {c[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {c[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {d[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {d[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {d[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {e[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {e[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {e[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {s1[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {s2[0]}]
```

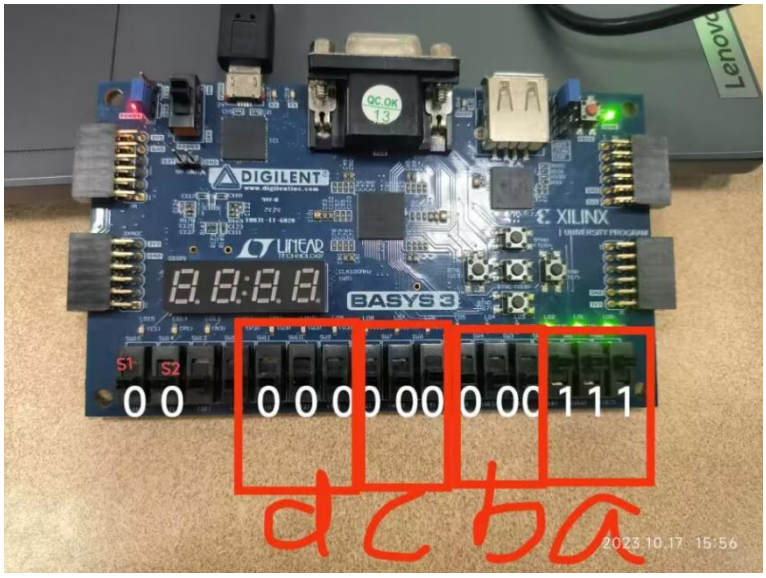
（5） 生成比特流，打开硬件程序并打开硬件目标，在开发板上验证自己的逻辑程序。



d=2’b110,c=2’b000,b=2’b000,a=2’b000,s1=1,s2=1;f=d=2’b110

d=2’b000,c=2’b100,b=2’b000,a=2’b000,s1=1,s2=0;f=c=2’b100

d=2’b000,c=2’b000,b=2’b101,a=2’b000,s1=0,s2=1;f=b=2’b101



d=2'b000,c=2'b000,b=2'b000,a=2'b111,s1=0,s2=0;f=d=2'b111

经过验证，与预期真值表和方法一的结果完全一致。

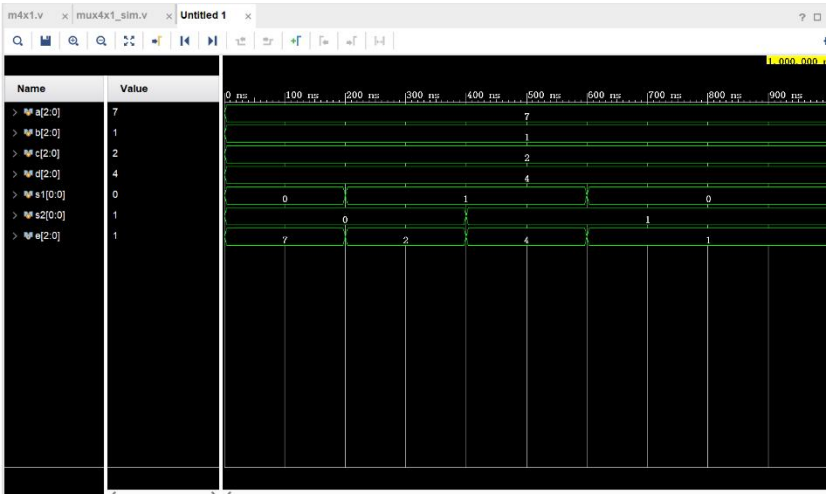
五、实验过程中遇到的问题及解决情况

在我们进行 block design 设计时，首先尝试着使用三个二选一选择器叠加实现四选一选择器，但是仿真结果输出的一直是 z，最后寻找出问题是没有考虑到输入位宽的问题。想要套用实验二设计的二选一选择器的 ip 核，但是忘记了输入的位宽和判断的位宽是不同的，于是重新编写二选一选择器，使用 if-else 语句很好的解决了这个问题，在不改变原始架构的条件下实现的三位宽选择。但是在进行四选一选择器设计时忘记设置输入的位宽，最后也是及时发现并解决了这个问题。

六、实验结果及分析和（或）源程序调试过程

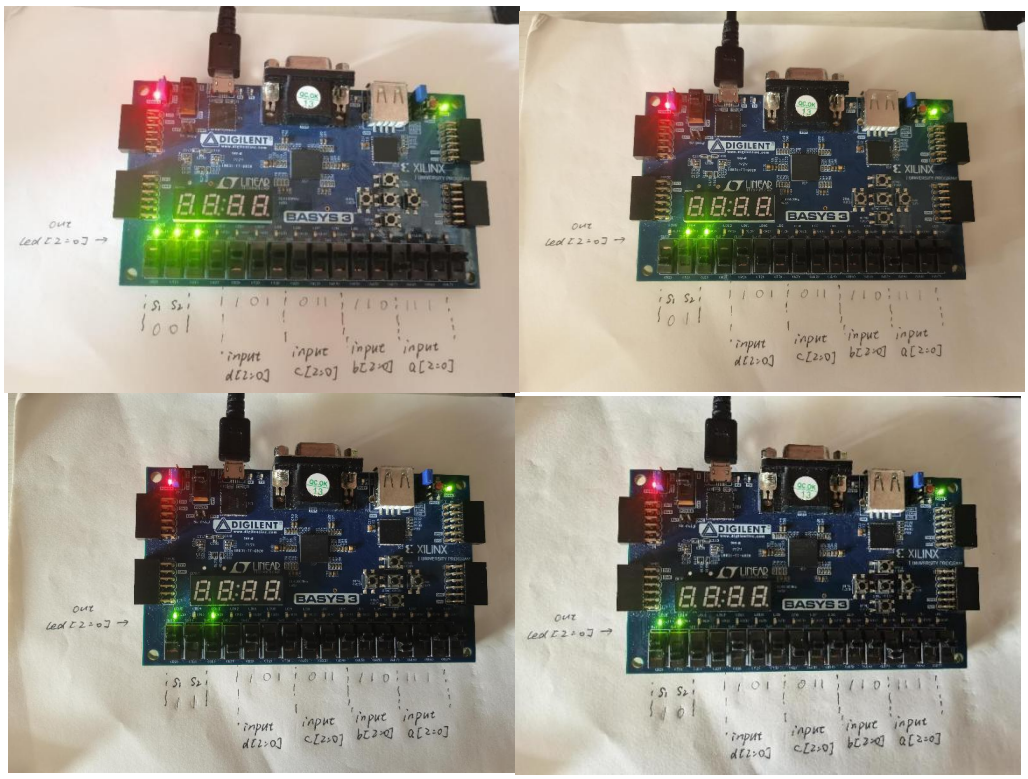
1.实验结果

（1）经过 block design 设计和 verilog 语言程序设计的设计文件在仿真和最后实现的过程中均得到了预期结果。



仿真结果

（2）在仿真测试成功后，完成综合、实现、引脚分配的过程，最后成功生成比特流并将其下载到开发板上。



在四位输入分别是 111,011,101,110 时，改变选择开关 s1,s2 时输出情况

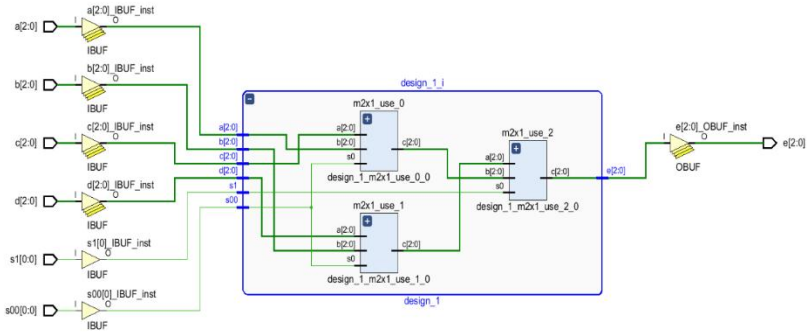
通过在开发板上进一步验证自己的程序，可以得出本实验通过 block

design 和 verilog 语言设计两种方式成功完成了 3 位的四选一多路选择器的实现。

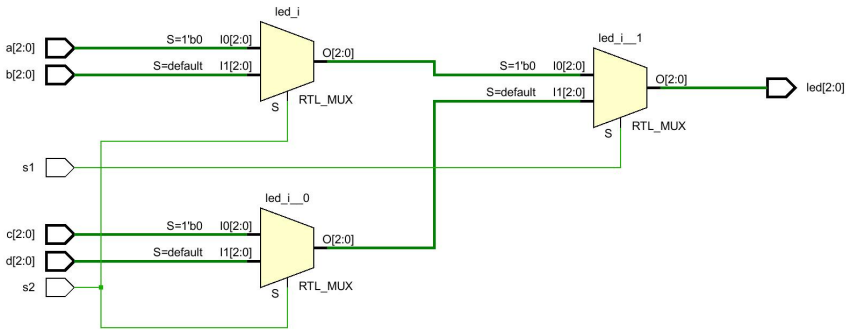
2.实验结果分析

经过 block design 设计和 verilog 语言设计实现同样功能的四选一多路选择器，其在具体实现过程中存在着怎样的不同，下面就 vivado 的 RTL 分析功能和功耗分析功能上对比分析。

（1）RTL 分析：



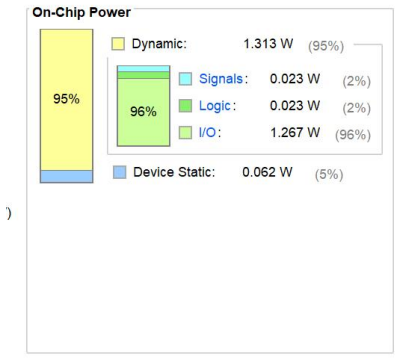
Block design 实现的 RTL 分析



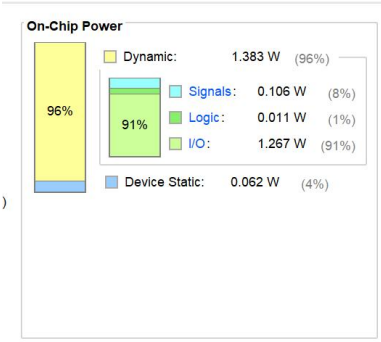
Verilog 语言设计实现的 RTL 分析

由上图可以看出，即使在两种不同实现方式在 RTL 分析中从封装到基本元件都基本相同，可以看出无论是通过 block design 实现还是 verilog 语言实现，vivado 等程序对其处理后得到的基本电路逻辑是相同的。因此在综合层级，block design 和 verilog 实现几乎没有区别。

（2）功耗和逻辑资源占用分析：



Block design 实现的功耗



verilog 实现的功耗

由上述对比可以看出，由于是在同一种开发板上开发，硬件所产生的功耗（device static）相同，均为 0.062W；而由于电路逻辑较为简单，本应出现差距的 I/O 功耗几乎相同；在 logic 和 signals 上出现了功耗的较大差距，据分析可能是实现过程中 block design 方式是调用 IP 核而 verilog 语言设计方式是调研开发班上本就有的集成逻辑，因此前者效率相对更慢。

七、小组分工情况说明

：完成 IP 核的编写和封装，编写 block design 和 verilog 语言设计的仿真代码，并进行仿真验证。共同分析对比二者在 RTL 分析、逻辑资源占用、功耗上的不同。

：利用封装的 IP 核完成 IP 核的调用，完成 block design 设计，并完成两种方式实现的 RTL 分析。共同分析对比二者在 RTL 分析、逻辑资源占用、功耗上的不同。

：编写通过 verilog 语言设计实现实验的设计文件代码，完成引脚绑定和上板验证。共同分析对比二者在 RTL 分析、逻辑资源占用、功耗上的不同。