同济大学计算机系

数字逻辑课程实验报告

学　　号 2252941

姓　　名 杨瑞灵

专　　业 计算机科学与技术

授课老师 张冬冬

一、实验内容

实验介绍：

在本次实验中，我们将使用 Verilog HDL 语言实现 4 位比较器、8 位比较器、无符号 加法器和有符号加法器的设计和仿真。

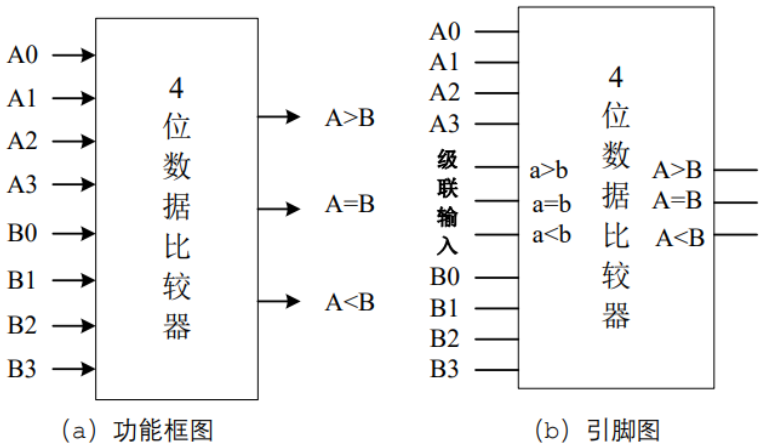
实验目标：

深入了解比较器和加法器的原理。
学习使用 Verilog HDL 语言设计 4 位比较器和 8 位比较器。
学习使用 Verilog HDL 语言设计无符号加法器和有符号加法器。

实验原理：

(1)6.5_1 四位数据比较器

用来完成两组二进制数大小比较的逻辑电路，称为数据比较器。图 6.5.1（a）给出了 为 4 位二进制数据比较器的功能框图。输入是一组二进制数 A3 A2 A1 A0（A3 为高位） 和另一组二进制数 B3 B2 B1 B0（B3 为高位），两组数比较的结果，只能是输出 A>B， A=B，A



（a）功能框图　　（b）引脚图

图 6.5.1 4 位 2 进制数比较器

表 6.5.1 4 位二进制数比较器 74C85 真值表

比较输入				级联输入			输出		
a3b3	a2b2	a1b1	a0b0	a>b	a<b	a=b	A>B	A<B	A=B
a3>b3	×	×	×	×	×	×	1	0	0
a3<b3	×	×	×	×	×	×	0	1	0
a3=b3	a2>b2	×	×	×	×	×	1	0	0
a3=b3	a2<b2	×	×	×	×	×	0	1	0
a3=b3	a2=b2	a1>b1	×	×	×	×	1	0	0
a3=b3	a2=b2	a1<b1	×	×	×	×	0	1	0
a3=b3	a2=b2	a1=b1	a0>b0	×	×	×	1	0	0
a3=b3	a2=b2	a1=b1	a0<b0	×	×	×	0	1	0
a3=b3	a2=b2	a1=b1	a0=b0	1	0	0	1	0	0
a3=b3	a2=b2	a1=b1	a0=b0	0	1	0	0	1	0
a3=b3	a2=b2	a1=b1	a0=b0	0	0	1	0	0	1

(2)6.5_2 八位数据比较器

当比较位数超过 4 位时，可以将两片或多片 74HC85 级联使用。如图 6.5.2 所示为 2 片 4 位比较器级联而成的 8 位比较器，此时低 4 位和高 4 位输入信号，分别加到两个比较器的输入端，低 4 位比较器的三个输出分别对应接到高 4 位比较器的三个级联输入端 a>b，a<b，a=b。比较结果由高 4 位比较器输出端输出。

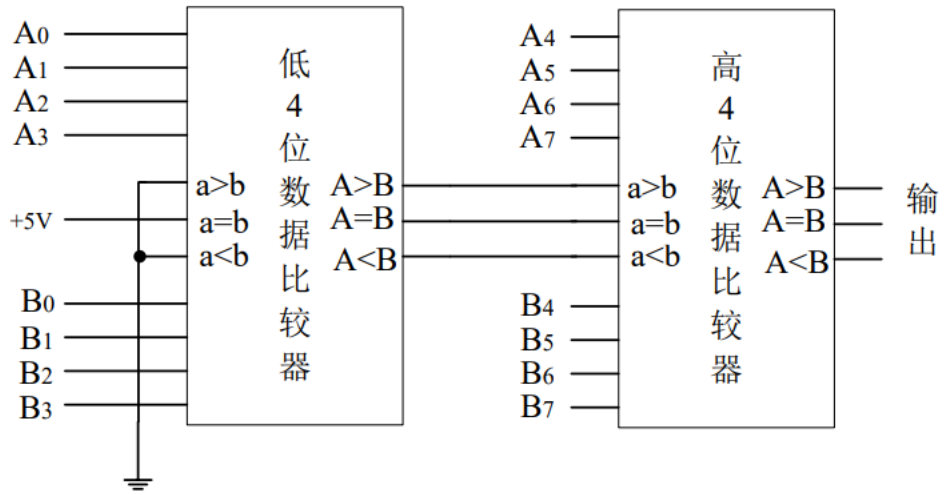


图 6.5.2 使用 2 片 74HC85 组成 8 位比较器

(3)6.5_3 加法器

加法器是计算机或其他数字系统中对二进制数进行运算处理的组合逻辑构件。本实验主 要采用 Verilog HDL 行为描述实现串行加法器，其逻辑结构框图如图 6.19（a）所示， 它由多个全加器（FA）串行连接而成。每一个全加器是一位加法器，其逻辑图如 6.19（b） 所示，有三个输入（加数Ai，被加数Bi，低位的进位信号Ci−1），两个输出（和数Si，向高 位的进位信号Ci）。

二、硬件逻辑图

三、模块建模

(1)6.5_1

```
module DataCompare4(
    input [3:0] iData_a,
    input [3:0] iData_b,
    input [2:0] iData,
    output reg[2:0] oData
    );
    always @(*)
    begin
    if(iData_a[3]&~iData_b[3]) oData=3'b100;
    else if(~iData_a[3]&iData_b[3]) oData=3'b010;
    else if(iData_a[2]&~iData_b[2]) oData=3'b100;
    else if(~iData_a[2]&iData_b[2]) oData=3'b010;
    else if(iData_a[1]&~iData_b[1]) oData=3'b100;
    else if(~iData_a[1]&iData_b[1]) oData=3'b010;
    else if(iData_a[0]&~iData_b[0]) oData=3'b100;
    else if(~iData_a[0]&iData_b[0]) oData=3'b010;
    else oData=iData;
    end
endmodule
```

(2)6.5_2

```
//四位比较器
module DataCompare4(
    input [3:0] iData_a,
    input [3:0] iData_b,
    input [2:0] iData,
    output reg[2:0] oData
    );
    always @(*)
    begin
    if(iData_a[3]&~iData_b[3]) oData=3'b100;
    else if(~iData_a[3]&iData_b[3]) oData=3'b010;
    else if(iData_a[2]&~iData_b[2]) oData=3'b100;
    else if(~iData_a[2]&iData_b[2]) oData=3'b010;
    else if(iData_a[1]&~iData_b[1]) oData=3'b100;
    else if(~iData_a[1]&iData_b[1]) oData=3'b010;
```

```
    else if(iData_a[0]&~iData_b[0]) oData=3'b100;
    else if(~iData_a[0]&iData_b[0]) oData=3'b010;
    else oData=iData;
    end
endmodule

//八位比较器
module DataCompare8(
    input [7:0] iData_a,
    input [7:0] iData_b,
    output [2:0] oData
    );
    //实例化两个四位比较器
    //低四位
    reg [3:0] iData_a_1;
    reg [3:0] iData_b_1;
    reg [2:0] iData_1;
    wire [2:0] oData_1;
    DataCompare4 dc4_1(iData_a_1,iData_b_1,iData_1,oData_1);
    //高四位
    reg [3:0] iData_a_2;
    reg [3:0] iData_b_2;
    reg [2:0] iData_2;
    wire [2:0] oData_2;
    DataCompare4 dc4_2(iData_a_2,iData_b_2,iData_2,oData_2);

    always @(*)
    begin
    iData_1<=3'b001;
    iData_a_1<=iData_a[3:0];
    iData_b_1<=iData_b[3:0];
    iData_a_2<=iData_a[7:4];
    iData_b_2<=iData_b[7:4];
    iData_2<=oData_1;
    end
    assign oData=oData_2;
endmodule
```

(3)6.5_3

```
module FA(
    input iA,
    input iB,
```

```
        input iC,
        output oS,
        output oC
);
wire t1,t2,t3;
xor x1(t1,iA,iB);
xor x2(oS,t1,iC);
and a1(t2,t1,iC);
and a2(t3,iA,iB);
or o1(oC,t2,t3);
endmodule

module Adder(
        input [7:0] iData_a,
        input [7:0] iData_b,
        input iC,
        output [7:0] oData,
        output oData_C
        );
        wire oC0,oC1,oC2,oC3,oC4,oC5,oC6;
        FA FA0(iData_a[0],iData_b[0],iC,oData[0],oC0);
        FA FA1(iData_a[1],iData_b[1],oC0,oData[1],oC1);
        FA FA2(iData_a[2],iData_b[2],oC1,oData[2],oC2);
        FA FA3(iData_a[3],iData_b[3],oC2,oData[3],oC3);
        FA FA4(iData_a[4],iData_b[4],oC3,oData[4],oC4);
        FA FA5(iData_a[5],iData_b[5],oC4,oData[5],oC5);
        FA FA6(iData_a[6],iData_b[6],oC5,oData[6],oC6);
        FA FA7(iData_a[7],iData_b[7],oC6,oData[7],oData_C);

endmodule
```

四、测试模块建模

(1)6.5_1

```
module DataCompare4_tb();
reg [3:0] iData_a;
reg [3:0] iData_b;
reg [2:0] iData;
wire [2:0] oData;
DataCompare4 dc4(iData_a,iData_b,iData,oData);
initial
```

```
begin
//大于 输出 100
iData<=4'b001;
iData_a<=4'b1010;
iData_b<=4'b0100;
//小于 输出 010
#50 iData_a<=4'b0010;
    iData_b<=4'b0100;
//等于 输出 001
#50 iData_a<=4'b0100;
    iData_b<=4'b0100;
//大于 输出 100
#50 iData<=4'b100;
//小于 输出 010
#50 iData<=4'b010;

end
endmodule
```

(2)6.5_2

```
module DataCompare8_tb();
reg [7:0] iData_a;
reg [7:0] iData_b;
wire [2:0] oData;
DataCompare8 dc4_1(iData_a,iData_b,oData);
initial
begin
//大于 输出 100
iData_a<=8'b1010_0000;
iData_b<=8'b0100_0000;
//小于 输出 010
#50 iData_a<=8'b0010_0000;
    iData_b<=8'b0100_0000;
//等于 输出 001
#50 iData_a<=8'b0100_0000;
    iData_b<=8'b0100_0000;
end
endmodule
```

(3)6.5_3

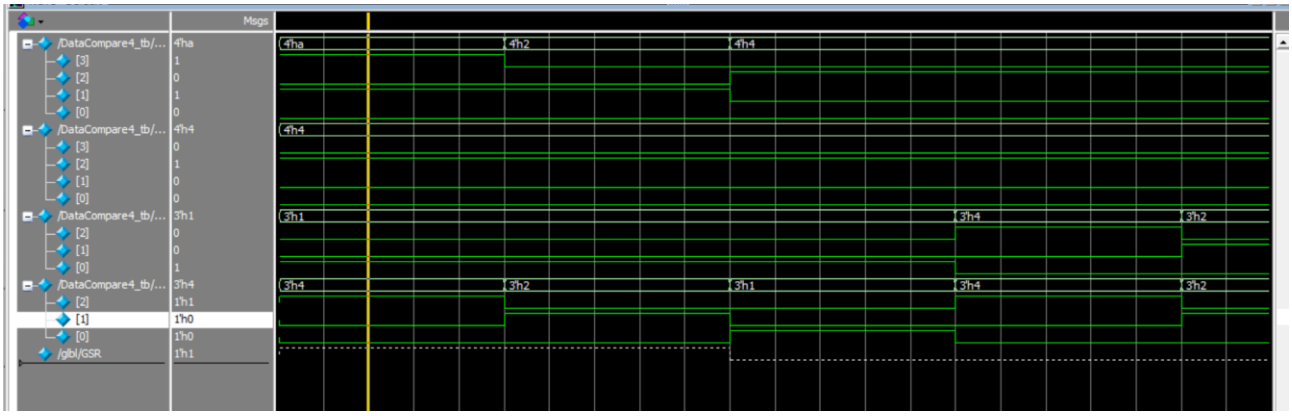
```
module Adder_tb();
reg [7:0] iData_a;
```

```
reg [7:0] iData_b;
reg iC;
wire [7:0] oData;
wire oData_C;
Adder adder(iData_a,iData_b,iC,oData,oData_C);
initial
begin
iC<=1'b1;
iData_a<=8'b1000_0001;
iData_b<=8'b1000_1111;
end
endmodule
```

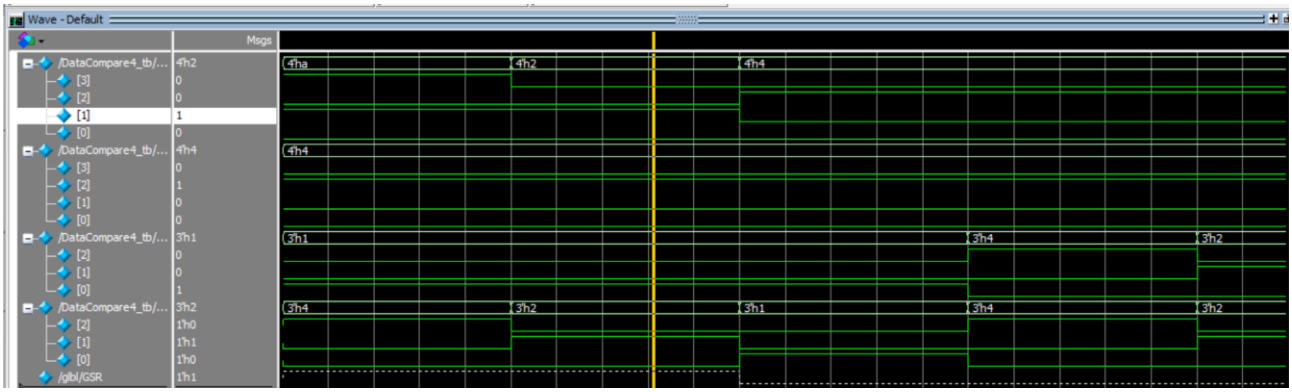
五、实验结果

(1)6.5_1

iData_a:1010
iData_b:0100
oData:100



iData_a:0010
iData_b:0100
oData:010

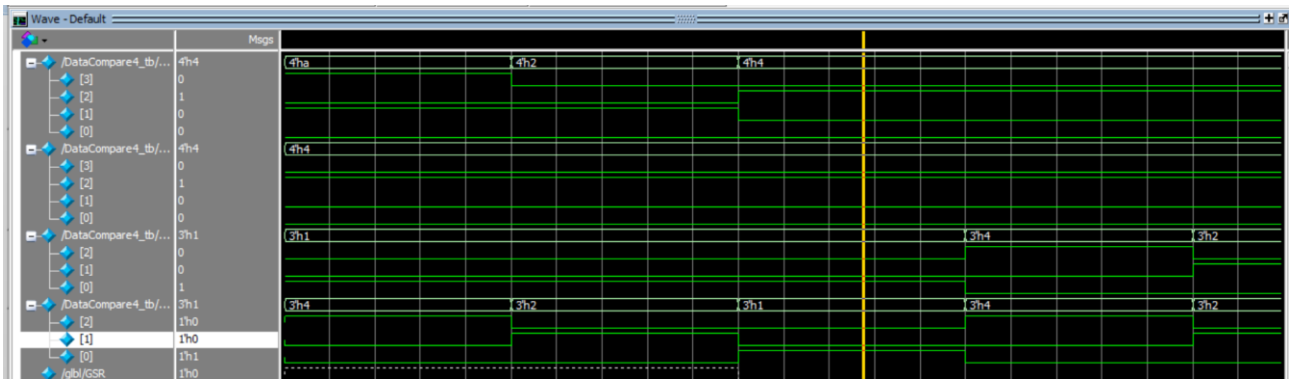


iData_a:0100

iData_b:0100

iData:001

oData:001

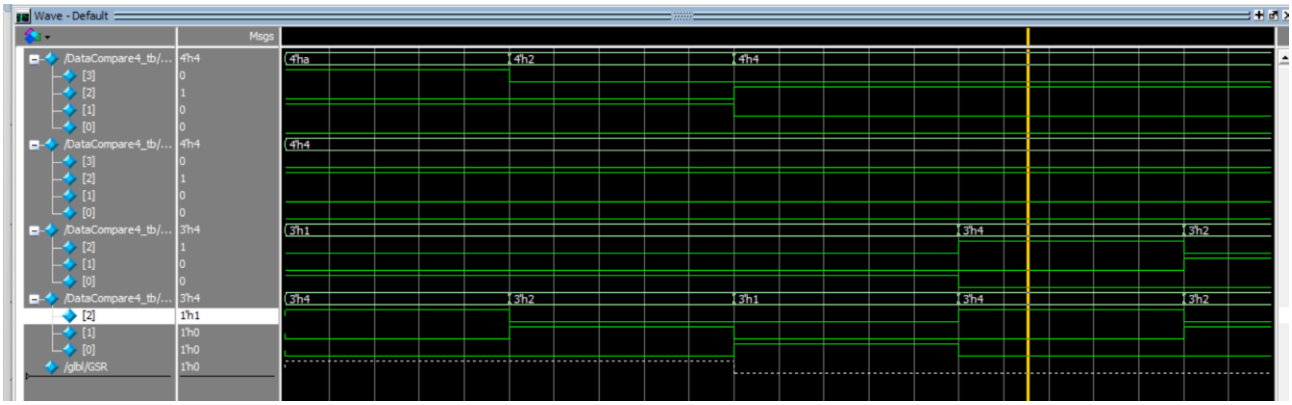


iData_a:0100

iData_b:0100

iData:100

oData:100

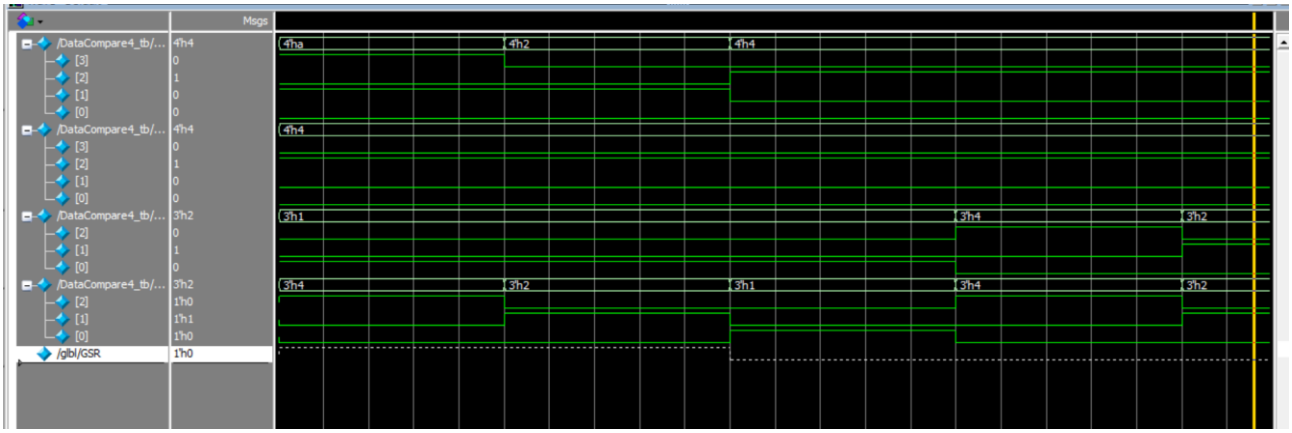


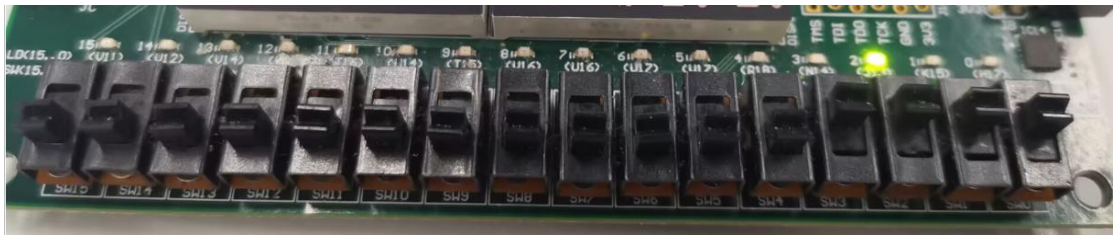
iData_a:0100

iData_b:0100

iData:010

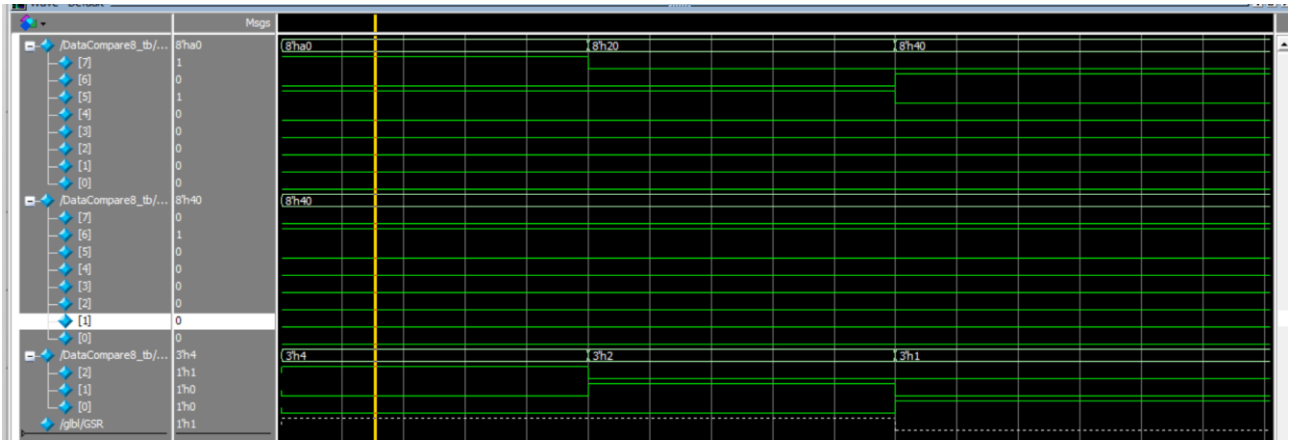
oData:010





(2)6.5_2

iData_a:1010 0000
iData_b:0100 0000
oData:100



iData_a:0010 0000

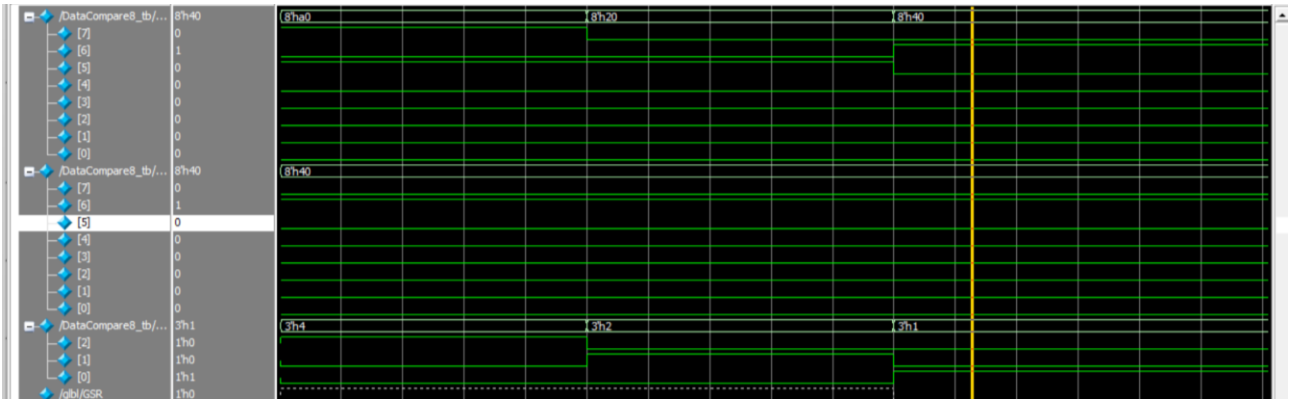
iData_b:0100 0000

oData:010

iData_a:0100 0000

iData_b:0100 0000

oData:001



(3)6.5_3

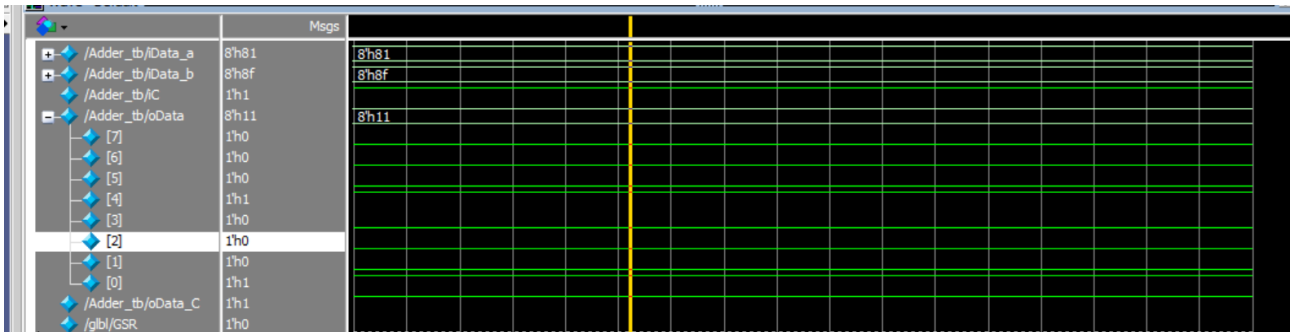
iData_a:1000 0001

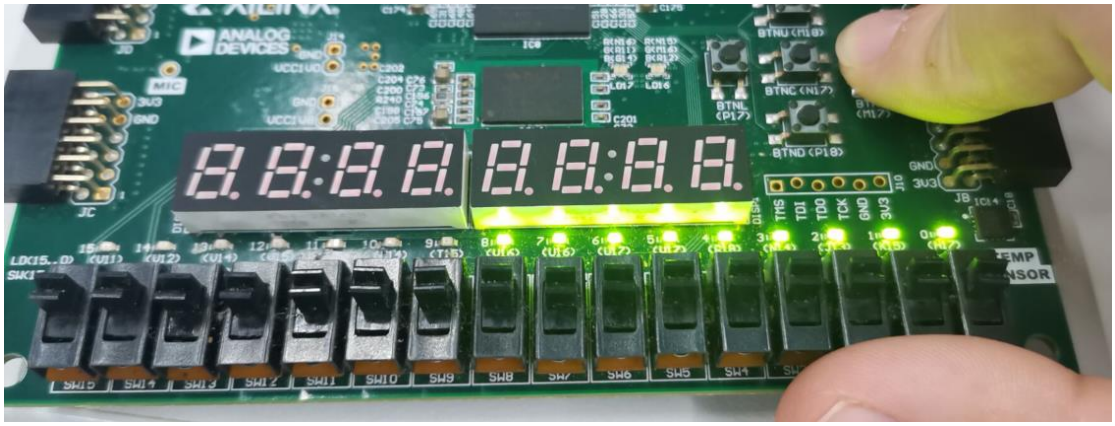
iData_b:1000 1111

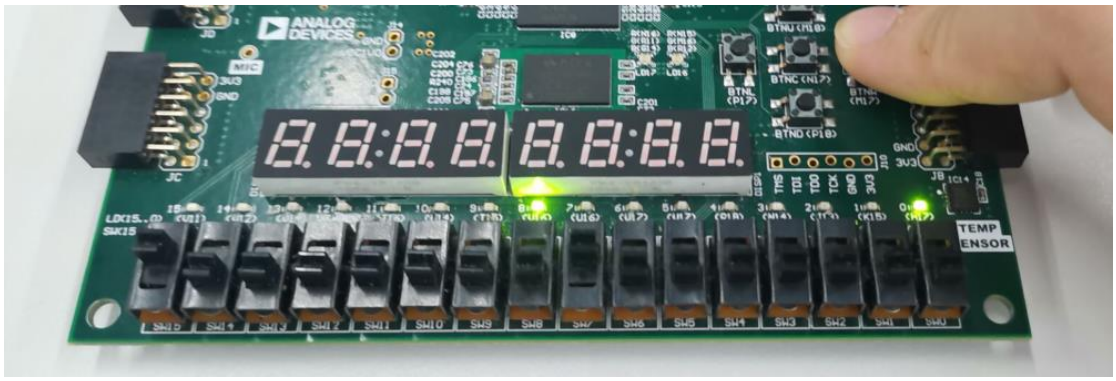
iC:1

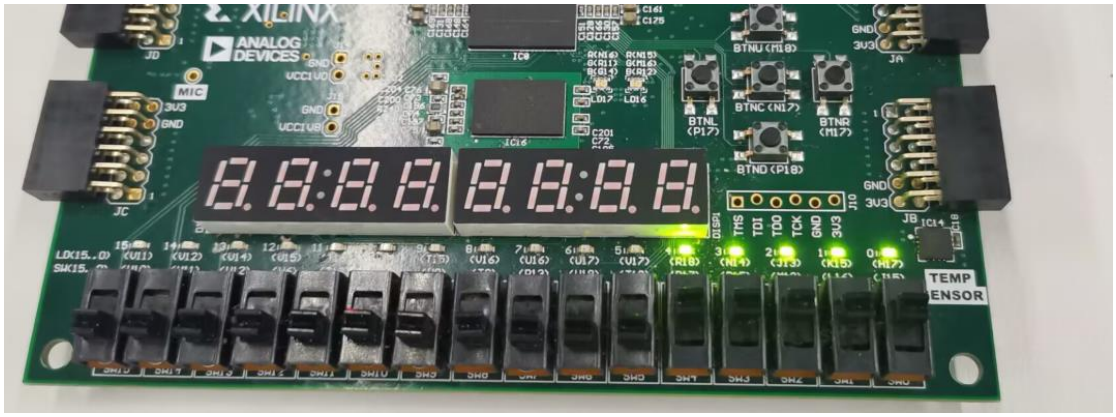
oData:0001 0001

oData_C:1




ANALOG DEVICES
XILINX
MIC
BTNU (M18)
BTNC (N17)
BTNL (P17)
BTND (P18)
TEMP SENSOR


ANALOG DEVICES
MIC
BTNU (M18)
BTNC (N17)
BTNL (P17)
BTND (P18)
TEMP SENSOR


XILINX
ANALOG DEVICES
MIC
BTNU (M18)
BTNC (N17)
BTNL (P17)
BTND (P18)
TEMP SENSOR