一、实验内容

在本次实验中，将使用 Verilog HDL 语言设计实现：

（1）4 位数据比较器的设计和仿真

（2）8 位数据比较器的设计和仿真

（3）1 位加法器的设计和仿真

（4）8 位串行加法器的设计和仿真

二、模块建模

（1）4 位数据比较器的设计和仿真

功能描述：完成两组 4 位二进制数的大小比较。

```
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]||(iData_a[3] ==iData_b[3]&&iData_a[2] > iData_b[2])||
              (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] > iData_b[1])||
              (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                                                  &&iData_a[0] > iData_b[0])||
              (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                  &&iData_a[0] == iData_b[0]&&iData[2]==1&&iData[1]==0&&iData[0]==0))
                oData = 3'b100;
            else if(iData_a[3] < iData_b[3]||(iData_a[3] ==iData_b[3]&&iData_a[2] <iData_b[2])||
                  (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] < iData_b[1])||
                  (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                                                  &&iData_a[0] < iData_b[0])||
                  (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                  &&iData_a[0] == iData_b[0]&&iData[2]==0&&iData[1]==1&&iData[0]==0))
                oData = 3'b010;
            else if((iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                  &&iData_a[0] == iData_b[0]&&iData[2]==0&&iData[1]==0&&iData[0]==1))
                oData = 3'b001;
            else
                oData= 3'b000;
         end

endmodule
```

（2）8 位数据比较器的设计和仿真

功能描述：通过实例化 4 位数据比较器，完成两组 8 位二进制数的大小比较。

```
module DataCompare8(
    input [7:0] iData_a,
    input [7:0] iData_b,
    output [2:0] oData
    );
        wire [2:0]zhongjianxian;
        DataCompare4 disiwei(
                .iData_a(iData_a[3:0]),
                .iData_b(iData_b[3:0]),
                .oData(zhongjianxian[2:0])
            );
         DataCompare4 gaosiwei(
                 .iData_a(iData_a[7:4]),
                 .iData_b(iData_b[7:4]),
                 .iData(zhongjianxian[2:0]),
                 .oData(oData[2:0])
            );
endmodule
```

```
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]||(iData_a[3] ==iData_b[3]&&iData_a[2] > iData_b[2])||
              (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] > iData_b[1])||
              (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                                                  &&iData_a[0] > iData_b[0])||
              (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                  &&iData_a[0] == iData_b[0]&&iData[2]==1&&iData[1]==0&&iData[0]==0))
                oData = 3'b100;
            else if(iData_a[3] < iData_b[3]||(iData_a[3] ==iData_b[3]&&iData_a[2] <iData_b[2])||
                  (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] < iData_b[1])||
                  (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                                                  &&iData_a[0] < iData_b[0])||
                  (iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                  &&iData_a[0] == iData_b[0]&&iData[2]==0&&iData[1]==1&&iData[0]==0))
                oData = 3'b010;
            else if((iData_a[3] ==iData_b[3]&&iData_a[2] == iData_b[2]&&iData_a[1] == iData_b[1]
                                  &&iData_a[0] == iData_b[0]&&iData[2]==0&&iData[1]==0&&iData[0]==1))
                oData = 3'b001;
            else
                oData= 3'b001;
         end
endmodule
```

（3）1 位加法器的设计和仿真

功能描述：对两组 1 位二进制数进行加法运算。

```
module FA(
    input iA,
    input iB,
    input iC,
    output   oS,
    output   oC
    );
    wire xor1,and1,and2;

    xor (xor1,iA,iB);
    xor (oS,xor1,iC);
    and(and1,iC,xor1);
    and(and2,iA,iB);
    or(oC,and1,and2);


endmodule
```

（4）8 位串行加法器的设计和仿真

功能描述：通过实例化调用 1 位加法器，实现计算两组 8 位二进制数。

```
module Adder(
    input [7:0] iData_a,
    input [7:0] iData_b,
    input iC,
    output [7:0] oData,
    output oData_C
    );
    wire l1,l2,l3,l4,l5,l6,l7;

    FA f1(
                .iA(iData_a[0]),
                .iB(iData_b[0]),
                .iC(iC),
                .oS(oData[0]),
                .oC(l1)
               );
     FA f2(
                 .iA(iData_a[1]),
                 .iB(iData_b[1]),
                 .iC(l1),
                 .oS(oData[1]),
                 .oC(l2)
                );
```

```
    FA f3(
                .iA(iData_a[2]),
                .iB(iData_b[2]),
                .iC(l2),
                .oS(oData[2]),
                .oC(l3)
                );
     FA f4(
                .iA(iData_a[3]),
                .iB(iData_b[3]),
                .iC(l3),
                .oS(oData[3]),
                .oC(l4)
                 );
     FA f5(
                 .iA(iData_a[4]),
                 .iB(iData_b[4]),
                 .iC(l4),
                 .oS(oData[4]),
                 .oC(l5)
                  );
      FA f6(
                 .iA(iData_a[5]),
                 .iB(iData_b[5]),
                 .iC(l5),
                 .oS(oData[5]),
                 .oC(l6)
                  );
     FA f7(
                  .iA(iData_a[6]),
                  .iB(iData_b[6]),
                  .iC(l6),
                  .oS(oData[6]),
                  .oC(l7)
                   );
     FA f8(
                  .iA(iData_a[7]),
                  .iB(iData_b[7]),
                  .iC(l7),
                  .oS(oData[7]),
                  .oC(oData_C)
                   );
endmodule
```

```
module FA(
    input iA,
    input iB,
    input iC,
    output   oS,
    output   oC
    );
    wire xor1,and1,and2;

    xor (xor1,iA,iB);
    xor (oS,xor1,iC);
    and(and1,iC,xor1);
    and(and2,iA,iB);
    or(oC,and1,and2);

endmodule
```

三、测试模块建模

（1）4 位数据比较器的设计和仿真

```
module DataCompare4_tb;
    reg [3:0] iData_a;
    reg [3:0] iData_b;
    reg [2:0] iData;
    wire [2:0] oData;

    DataCompare4 uut (
        .iData_a(iData_a),
        .iData_b(iData_b),
        .iData(iData),
        .oData(oData)
    );

    initial begin
        // 测试 1: iData_a 大于 iData_b
        iData_a = 4'b0101; iData_b = 4'b0011; iData = 3'b000;
         #10;
        // 测试 2: iData_a 小于 iData_b
        iData_a = 4'b0011; iData_b = 4'b0101; iData = 3'b000;
         #10;
        // 测试 3: iData_a 等于 iData_b
        iData_a = 4'b0101; iData_b = 4'b0101; iData = 3'b100;
         #10;
        // 测试 4: 相等时的特殊判断
        iData_a = 4'b0101; iData_b = 4'b0101; iData = 3'b010;
```

```
        #10;
        // 测试 5: 另一个 iData 的特殊条件（假设这里表示某种特殊情况下的 iData_a 小于 iData_b）
        iData_a = 4'b0101; iData_b = 4'b0101; iData = 3'b001;
        #10;
    end
endmodule
```

（2）8 位数据比较器的设计和仿真

```
module DataCompare8_tb;
    reg [7:0] iData_a;
    reg [7:0] iData_b;
    wire [2:0] oData;

    DataCompare8 uut (
        .iData_a(iData_a),
        .iData_b(iData_b),
        .oData(oData)
    );

    initial begin
        // 测试 1: iData_a 大于 iData_b
        iData_a = 8'b0101_0100; iData_b = 8'b0011_0010;
        #10;
        // 测试 2: iData_a 小于 iData_b
        iData_a = 8'b0011_0101; iData_b = 8'b0101_0111;
        #10;
        // 测试 3: iData_a 等于 iData_b
        iData_a = 8'b0101_0000; iData_b = 8'b0101_0000;
        #10;
    end
endmodule
```

（3）1 位加法器的设计和仿真

```
module FA_tb;
    reg iA;
    reg iB;
    reg iC;

    wire oS;
    wire oC;

    FA uut (
        .iA(iA),
        .iB(iB),
        .iC(iC),
```

```
        .oS(oS),
        .oC(oC)
    );
        initial begin

            iA = 0; iB = 0; iC = 0;
            #10; // 0 + 0 + 0 = 0, 进位 0
            iA = 0; iB = 0; iC = 1;
            #10; // 0 + 0 + 1 = 1, 进位 0
            iA = 0; iB = 1; iC = 0;
             #10; // 0 + 1 + 0 = 1, 进位 0
            iA = 0; iB = 1; iC = 1;
             #10; // 0 + 1 + 1 = 0, 进位 1
            iA = 1; iB = 0; iC = 0;
            #10; // 1 + 0 + 0 = 1, 进位 0
            iA = 1; iB = 0; iC = 1;
             #10; // 1 + 0 + 1 = 0, 进位 1
            iA = 1; iB = 1; iC = 0;
            #10; // 1 + 1 + 0 = 0, 进位 1
            iA = 1; iB = 1; iC = 1;
             #10; // 1 + 1 + 1 = 1, 进位 1

        end
    endmodule
```

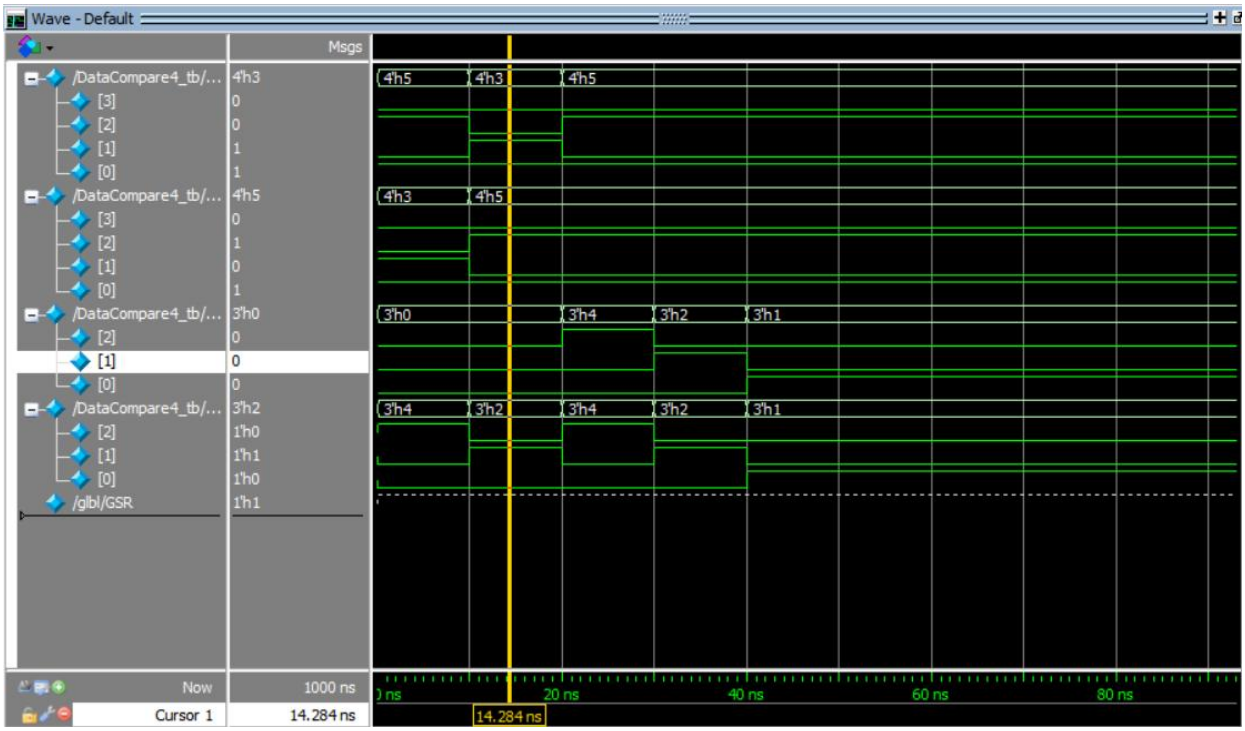
（4）8 位串行加法器的设计和仿真

```
module Adder_tb;

        reg [7:0] iData_a;
        reg [7:0] iData_b;
        reg iC;

        wire [7:0] oData;
        wire oData_C;

        Adder uut (
            .iData_a(iData_a),
            .iData_b(iData_b),
            .iC(iC),
            .oData(oData),
            .oData_C(oData_C)
        );
       initial begin
             iData_a = 8'h00; iData_b = 8'h00; iC = 0; #10; // 0 + 0 + 0 = 0, 进位 0
            iData_a = 8'h01; iData_b = 8'h01; iC = 0; #10; // 1 + 1 + 0 = 2, 进位 0
            iData_a = 8'h7F; iData_b = 8'h01; iC = 0; #10; // 127 + 1 + 0 = 128, 进位 1
            iData_a = 8'hFF; iData_b = 8'h01; iC = 0; #10; // 255 + 1 + 0 = 0 (8 位溢出), 进位 1
            iData_a = 8'hFF; iData_b = 8'hFF; iC = 0; #10; //1111 1110
        end
    endmodule
```

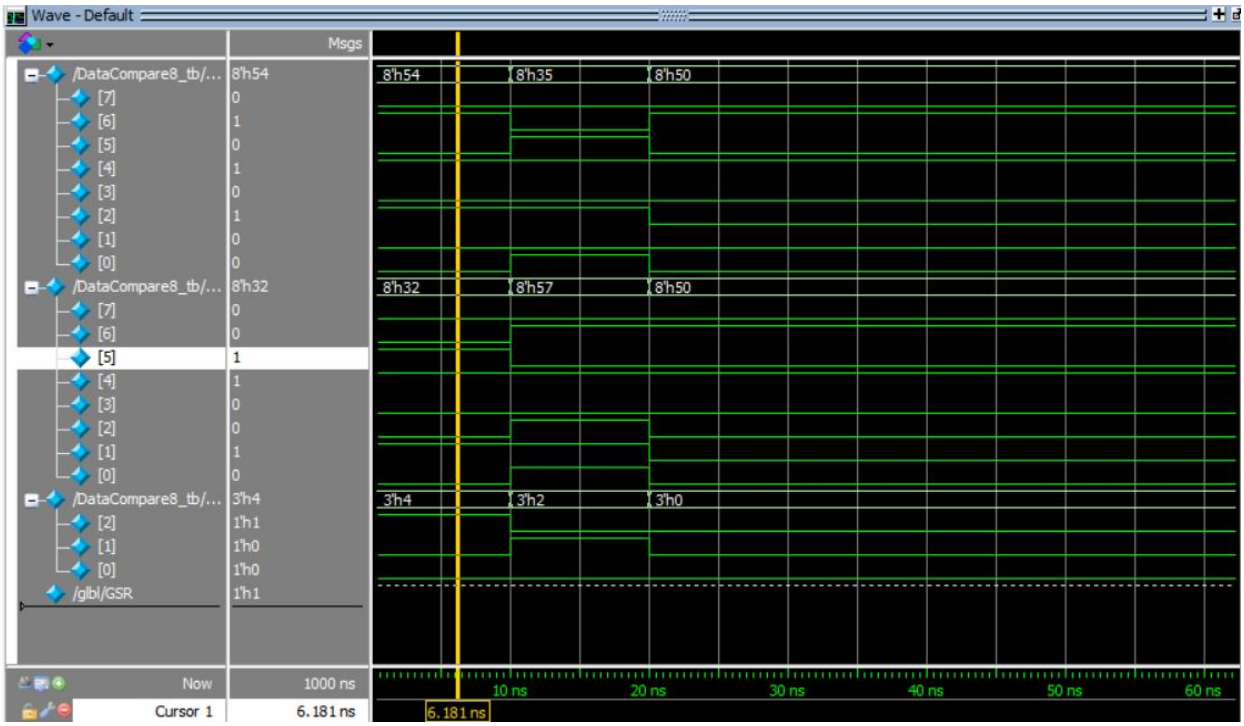
四、实验结果

1. modelsim 仿真波形图

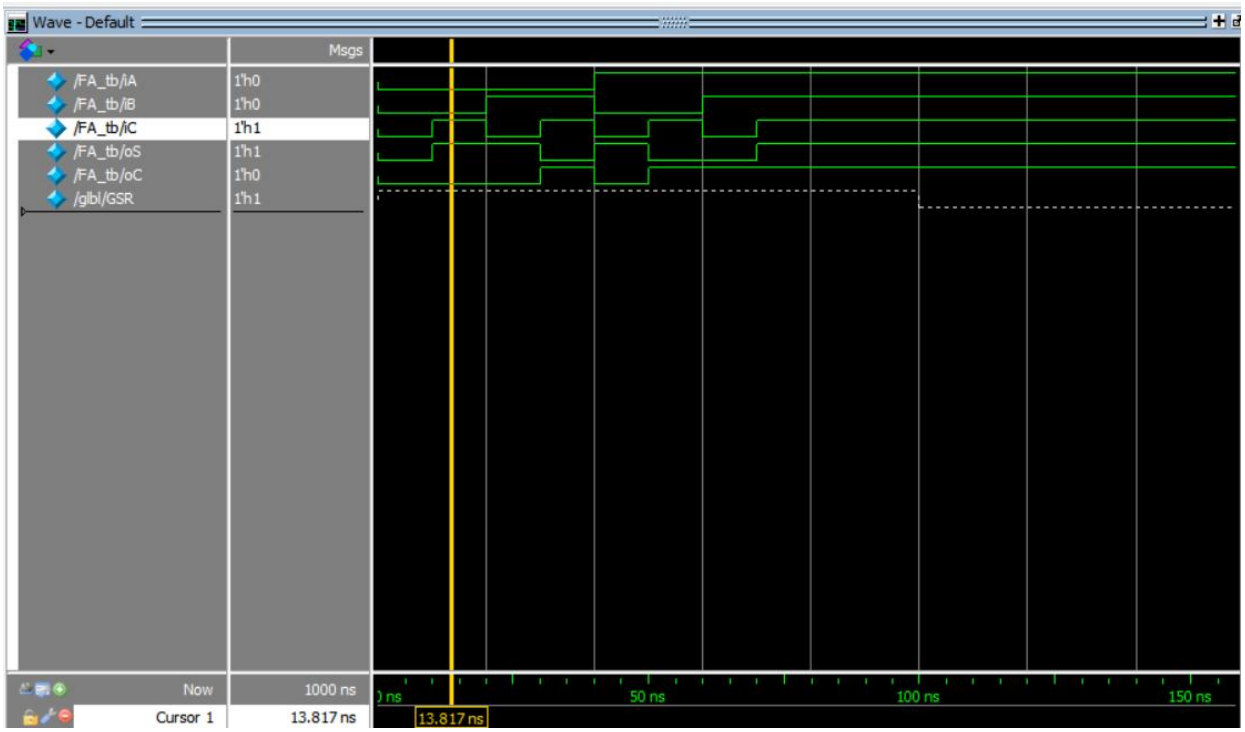
（1）4 位数据比较器的设计和仿真



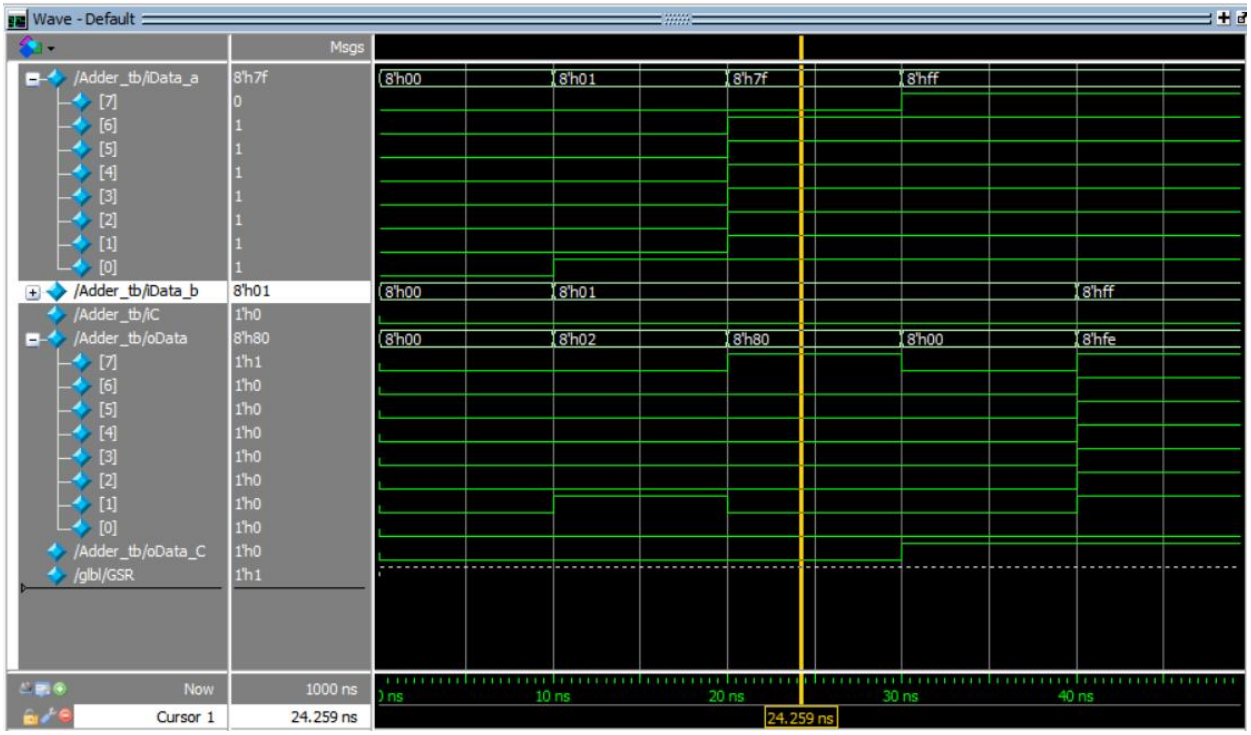
（2）8 位数据比较器的设计和仿真



（3）1 位加法器的设计和仿真

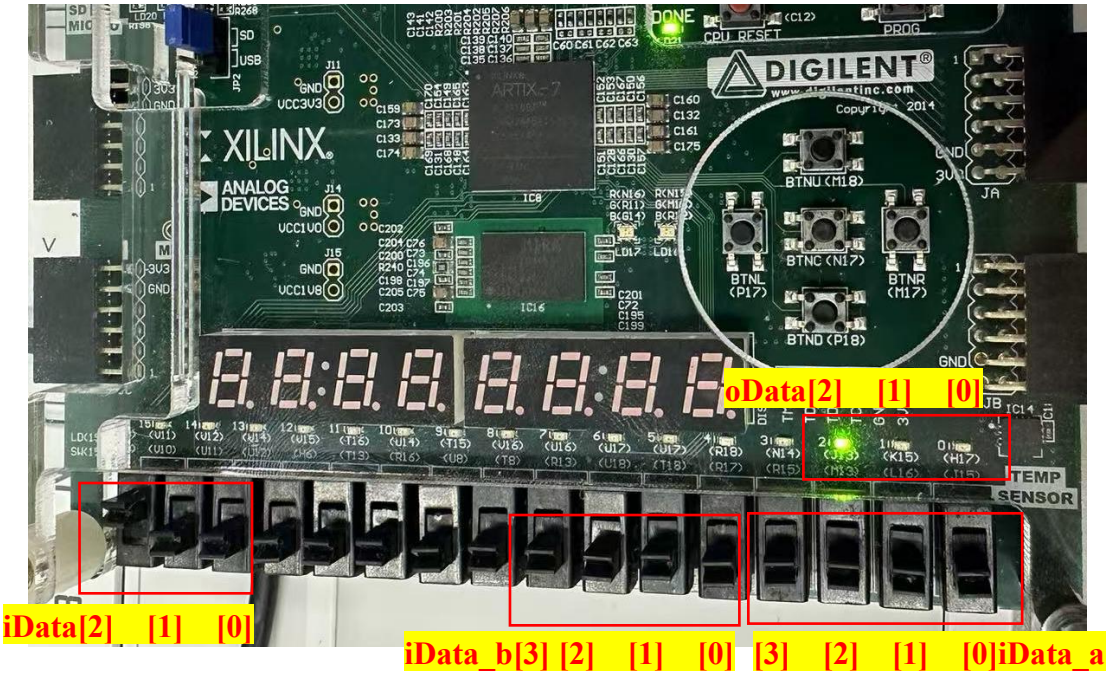


（4）8 位串行加法器的设计和仿真

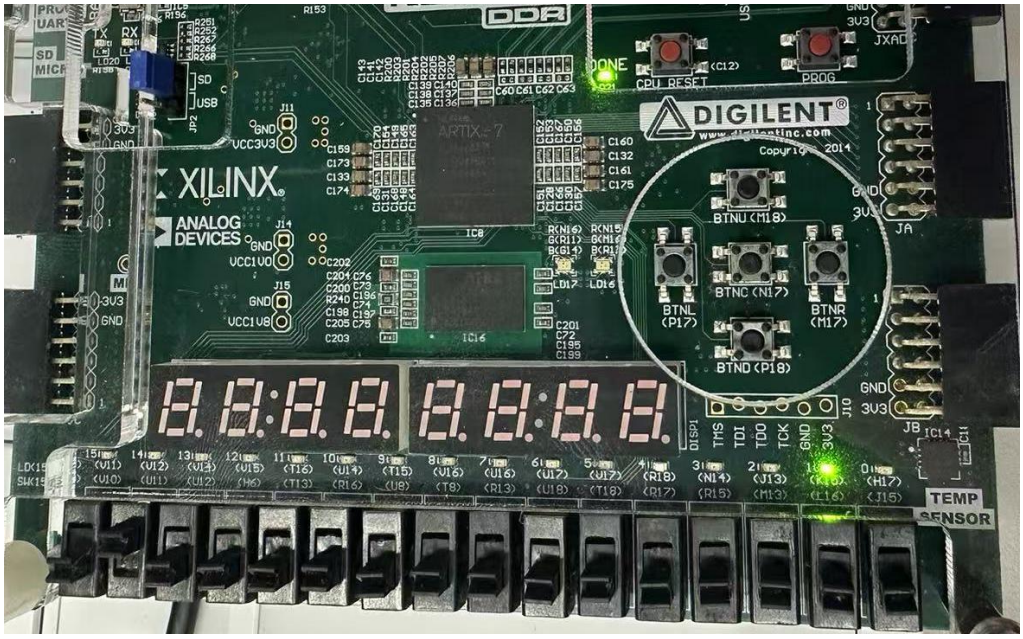


2. 下板后的实验贴图

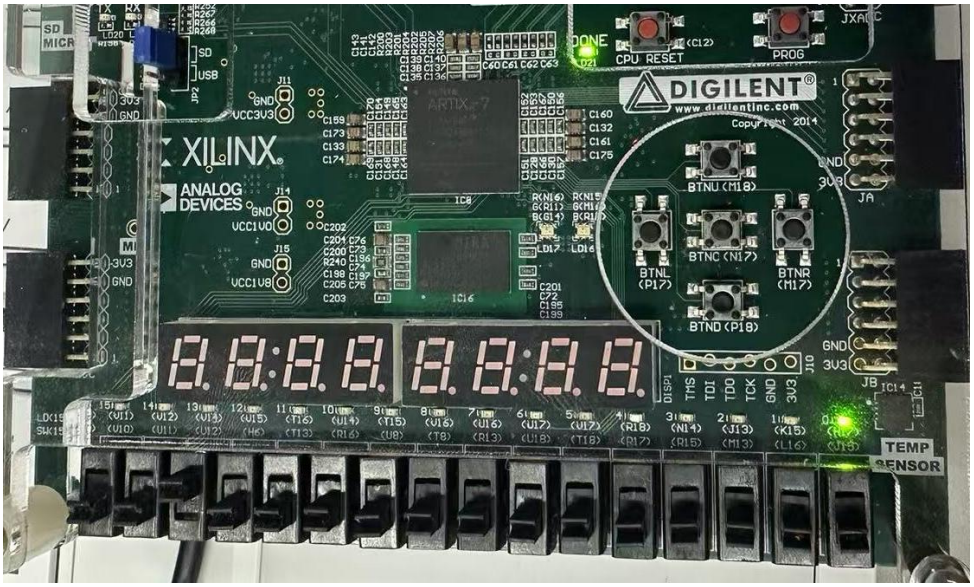
（1）4 位数据比较器的设计和仿真



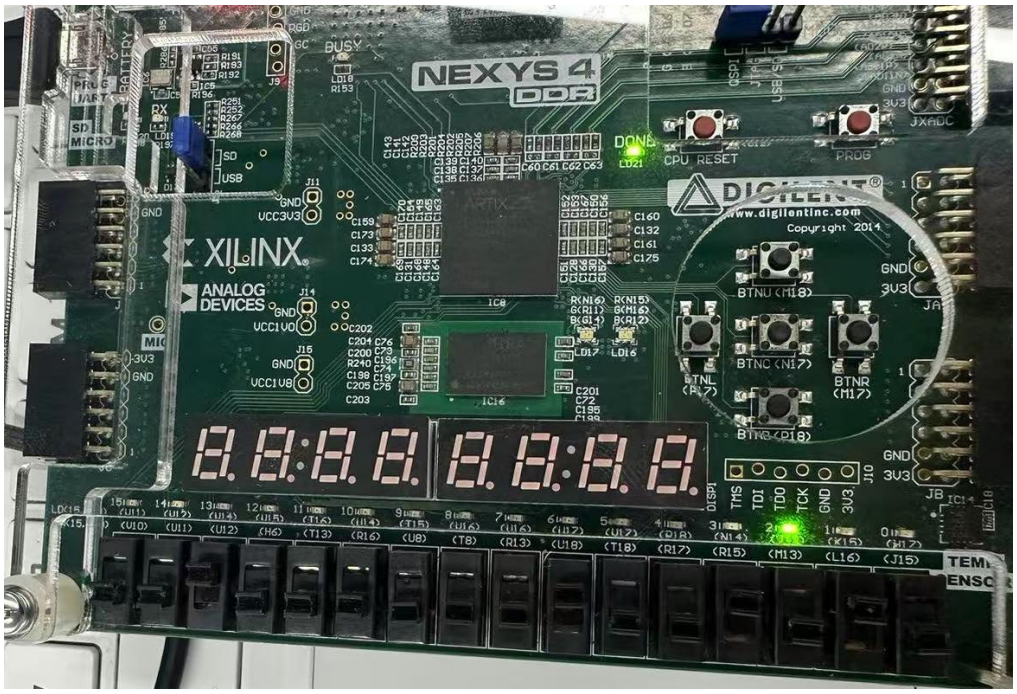
iData_a=iData_ b=0000 iData=100 ==>oData=100



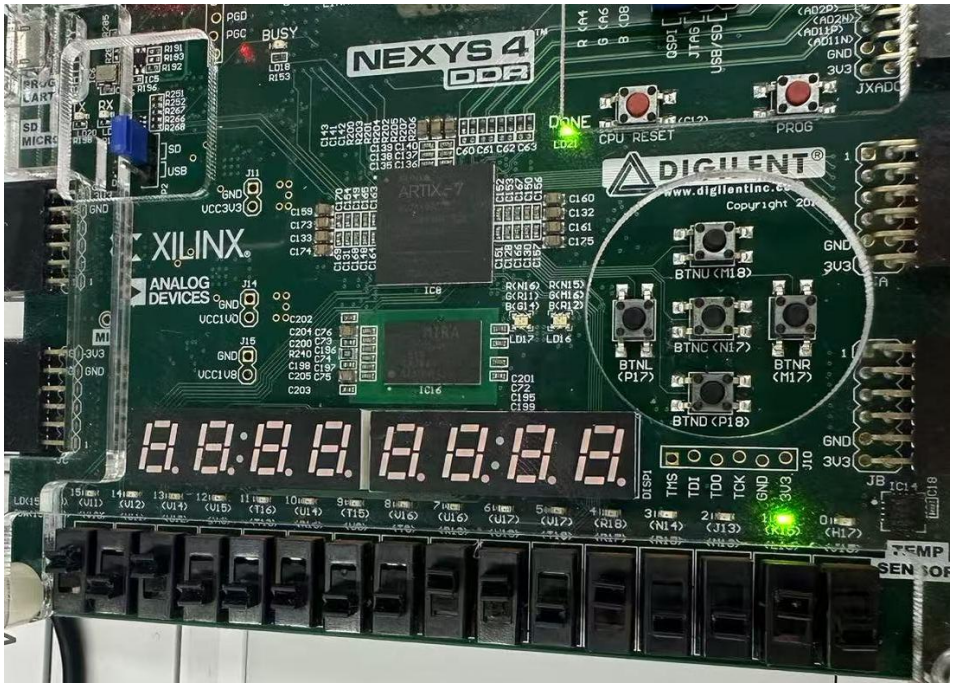
iData_a=iData_ b=0000 iData=010 ==>oData=010



iData_a=iData_ b=0000 iData=001 ==>oData=001

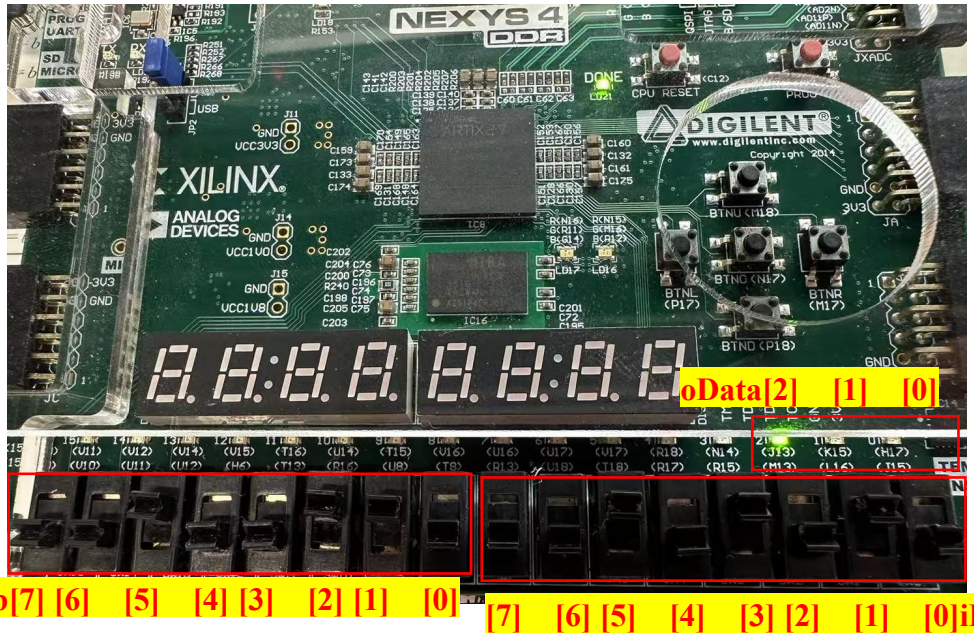


iData_a= 1010 iData_ b=0100 iData=001 ==>oData=100

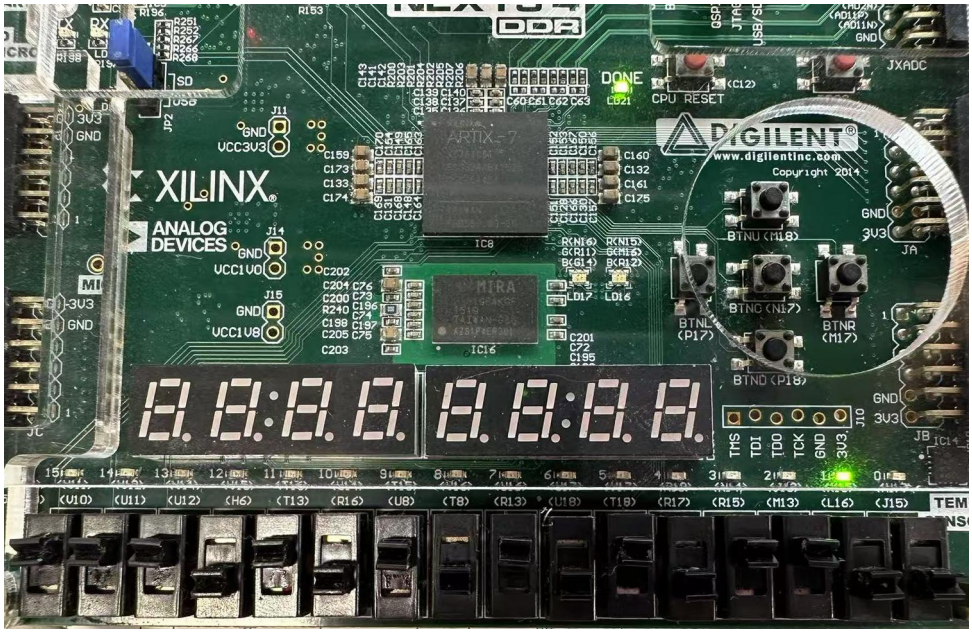


iData_a= 0010　iData_ b=1101　iData=101 ==>oData=010

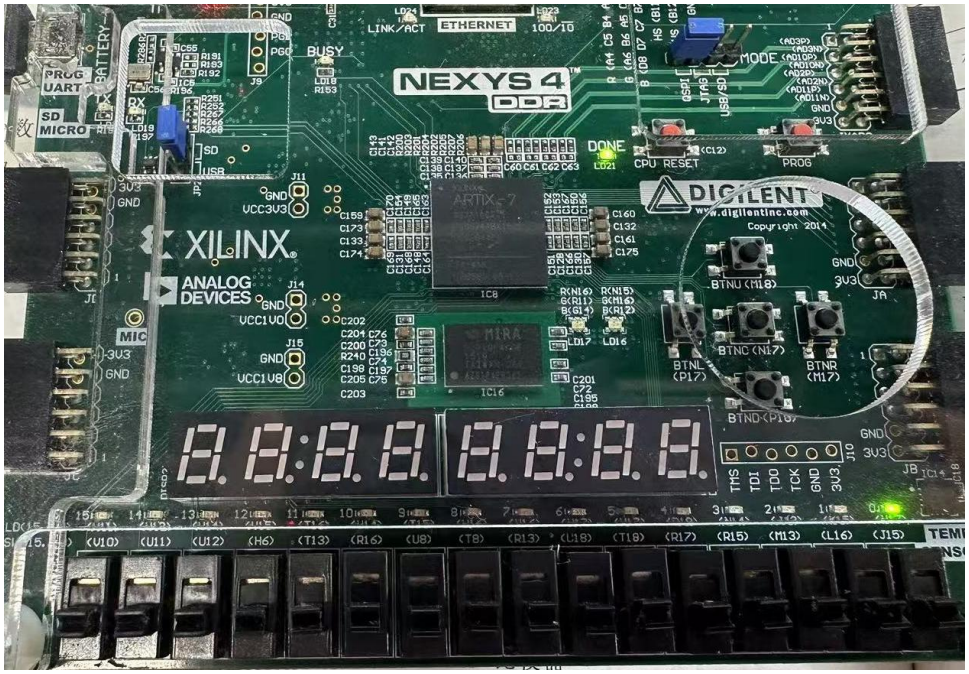
（2）8 位数据比较器的设计和仿真



iData_a= 0010 1010　iData_ b=0010 0110　==>oData=100

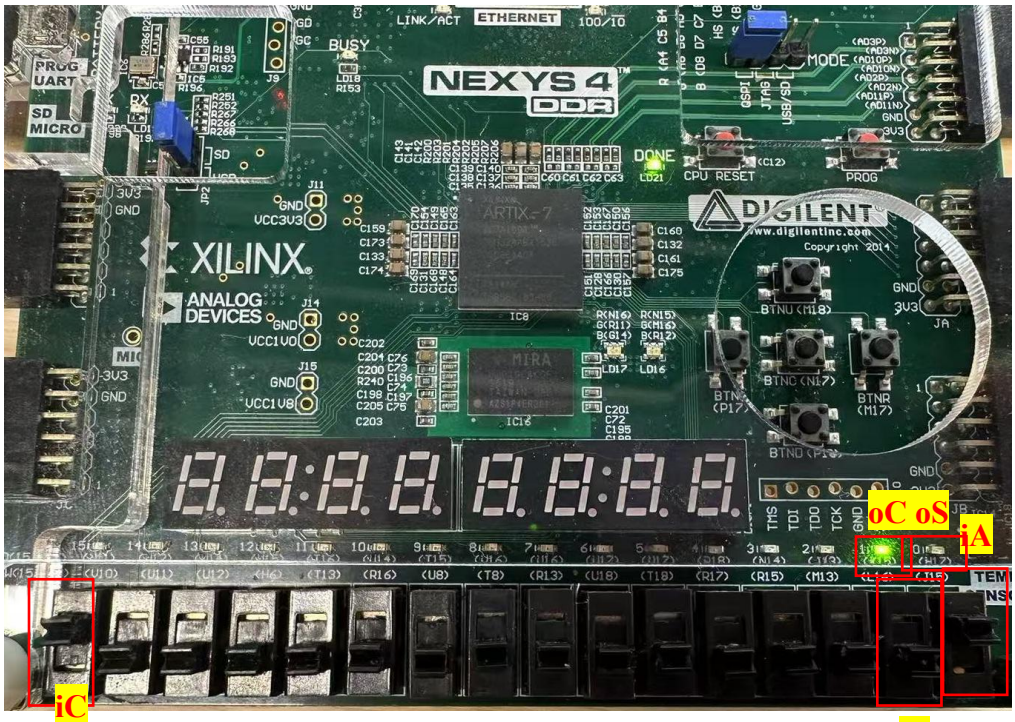


iData_a= 1010 0011 iData_ b=1110 1010 ==>oData=010

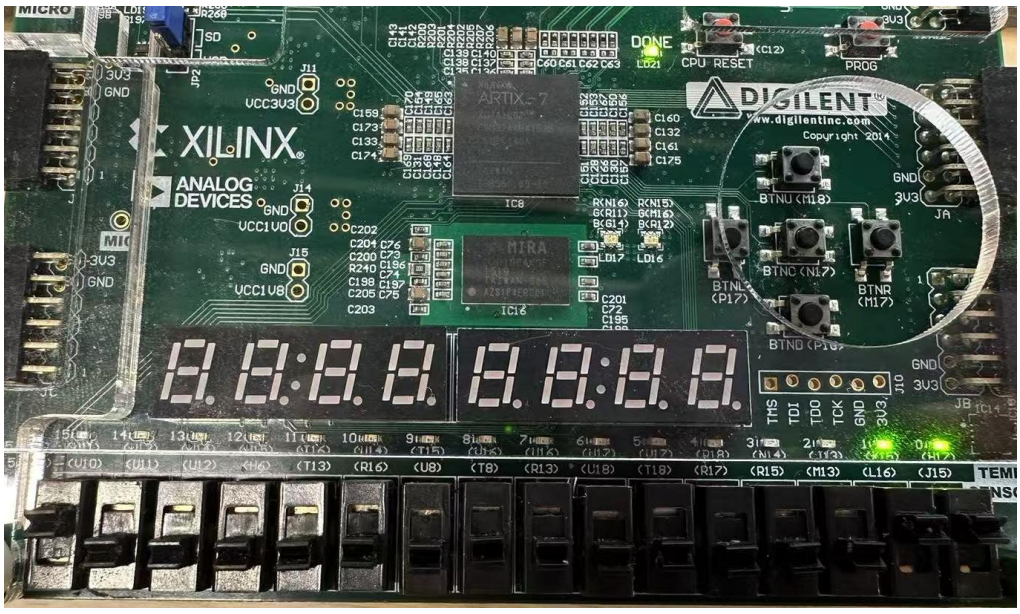


iData_a= iData_ b=0000 0000 ==>oData=001

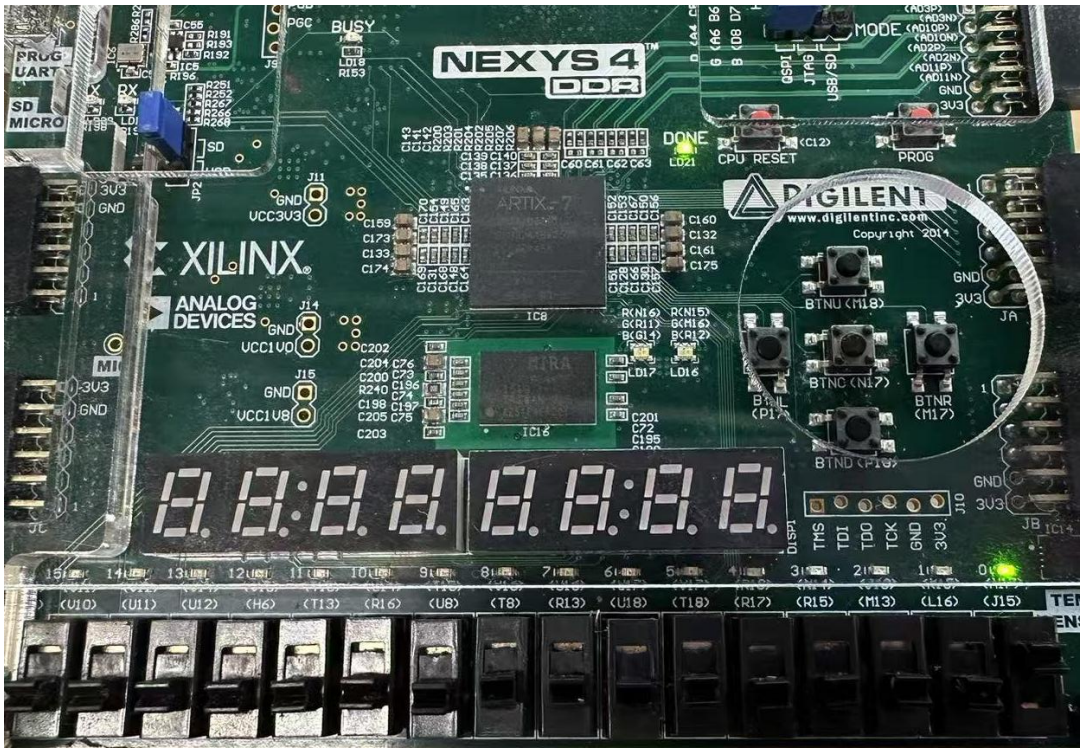
（3）1 位加法器的设计和仿真



iA=1 iB=0 iC=1 ==>oC=1 oS=0

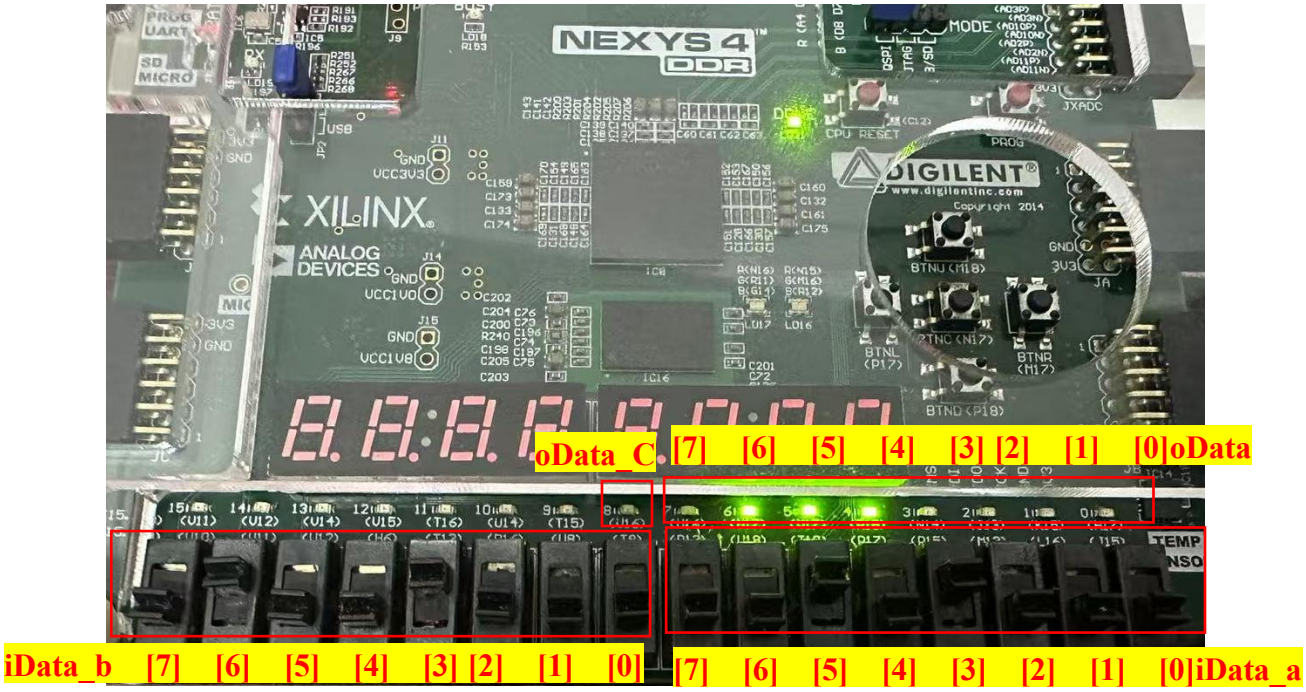


iA=1 iB=1 iC=1 ==>oC=1 oS=1

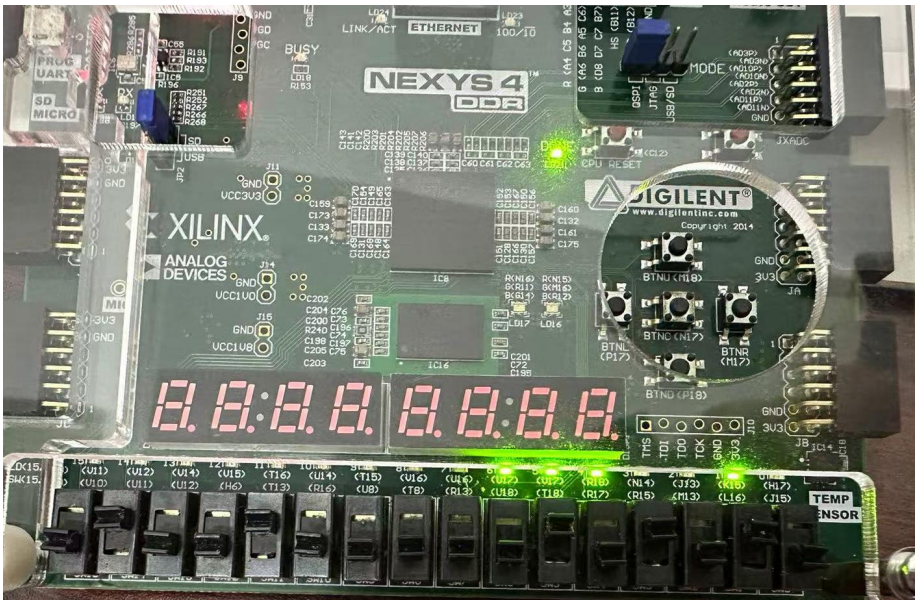


iA=1 iB=0 iC=0 ==>oC=0 oS=1

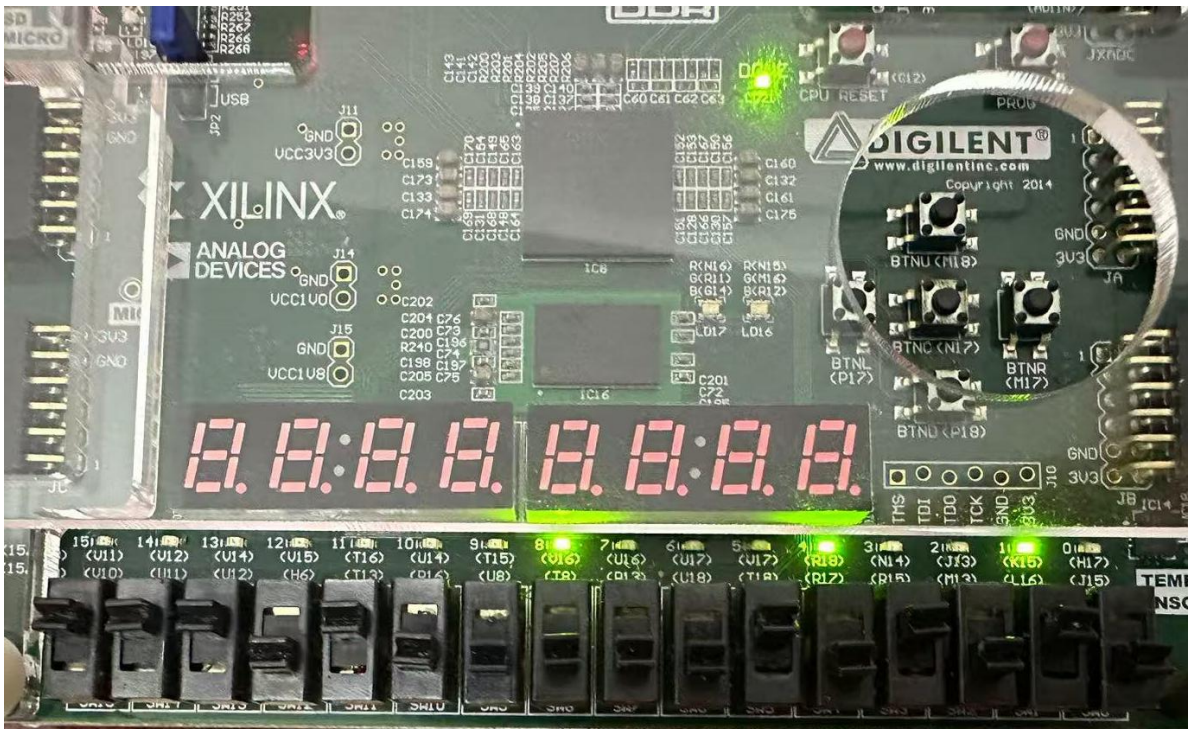
（4）8 位串行加法器的设计和仿真



iData_a=0010 1000 iData_b=0100 1000==>oData=0000 1110 oC=0



iData_a=0010 1010 iData_b=0100 1000==>oData=0111 0010 oC=0



iData_a=0010 1010 iData_b=1110 1000==>oData=0001 0010 oC=1