实验报告

实验题目：Lab 2　　　　日期：2018 年 10 月 19 日

姓名:刘紫檀　　　学号:PB17000232　　成绩:____________

实验目的：

1-1 以十进制定义学号最后一位，并在数码管上显示，提供代码及下载照片（1 张）——2 分

2-1　提供源代码、仿真截图及下载照片（0~9,10~15 两种情况各一张照片）——4 分

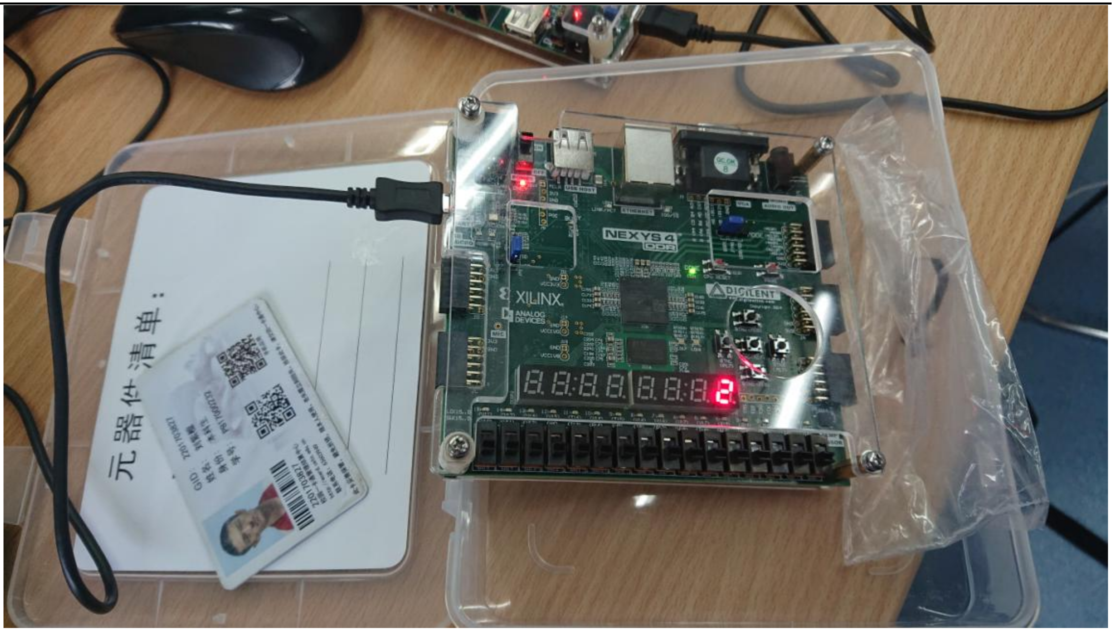
3-1 提供源代码、仿真截图及下载验证 照片，仿真、下载时将自己学号的最后两位作为输入，超出 7 的数字对 7 取余，并将进位位置为 1，检查截图并现场演示——4 分

所有下载照片中都应包含自己的一卡通

截图：

//实验 2-1-1 代码

```
`timescale 1ns / 1ps
module bcd_decoder(
    input [3:0] x,
    output reg [7:0] an,
    output reg [6:0] seg
    );

always @ (*)
begin
    an = 8'b11111110;
    case (x)
        4'd0: seg = 7'b1000000;
        4'd1: seg = 7'b1111001;
        4'd2: seg = 7'b0100100;
        4'd3: seg = 7'b0110000;
        4'd4: seg = 7'b0011001;
        4'd5: seg = 7'b0010010;
        4'd6: seg = 7'b0000010;
        4'd7: seg = 7'b1111000;
        4'd8: seg = 7'b0000000;
        4'd9: seg = 7'b0010000;
    endcase

end
endmodule

module main(
    output [7:0] an,
    output [6:0] seg
    );
    bcd_decoder(4'd2,an,seg);
endmodule
```



实验 2-1-1 下载照片

//实验 2-2-1 源代码， comparator.v

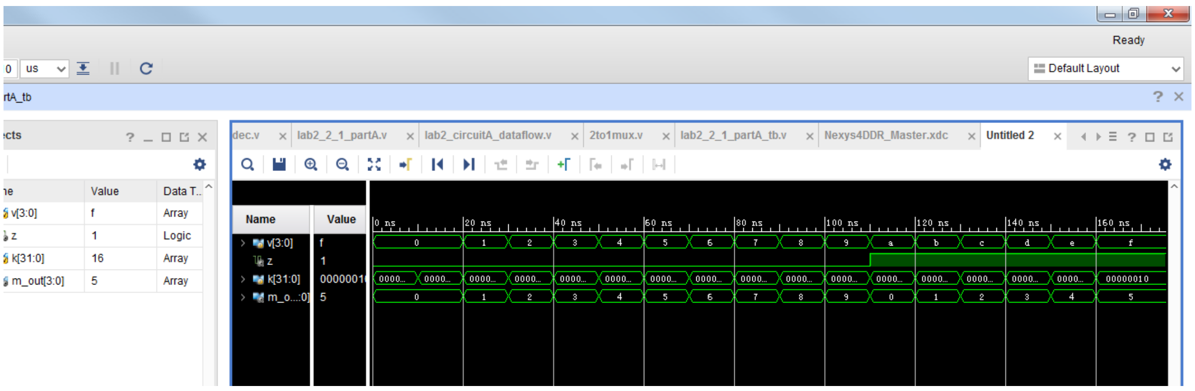
```
`timescale 1ns / 1ps

// x>y => res = 1
module comparator(
    input [3:0] x,
    input [3:0] y,
    output reg less,
    output reg equal,
    output reg greater
    );

    always @ (*)
    begin
        if (x > y)
        begin
           less = 0;
           equal = 0;
           greater = 1;
        end
        else if (x == y)
        begin
            less = 0;
            equal = 1;
            greater = 0;
        end
        else
        begin
            less = 1;
            equal = 0;
            greater = 0;
        end
    end
endmodule
```

//实验 2-2-1 代码， lab2_circuitA_dataflow.v

```
`timescale 1ns / 1ps

// -9 for 10~15
module lab2_circuitA_dataflow(
    input [3:0] vi,
    output [2:0] vo
    );

    assign vo[2] = (vi[3] & vi[2] & vi[1] & ~vi[0]) | (vi[3] & vi[2] & vi[1] & vi[0]);
    assign vo[1] = (vi[3] & vi[2] & ~vi[1] & ~vi[0]) | (vi[3] & vi[2] & ~vi[1] &
vi[0]);
    assign vo[0] = (vi[3] & ~vi[2] & vi[1] & vi[0]) | (vi[3] & vi[2] & ~vi[1] & vi[0])
|( vi[3] & vi[2] & vi[1] & vi[0]);
endmodule
```

//实验 2-2-1 代码， lab2_2_1_partA.v

```
`timescale 1ns / 1ps

module lab2_2_1_partA(
    input [3:0] v,
    output z,
    output [3:0] m
    );
    wire l,e,g;
    wire [3:0] voa;
    comparator cmp(v, 4'd9, l, e, z);
    lab2_circuitA_dataflow dfw(v, {voa[2], voa[1], voa[0]});
    Mux_2to1 mx1(v[3], 0, z, m[3]);
    Mux_2to1 mx2(v[2], voa[2], z, m[2]);
    Mux_2to1 mx3(v[1], voa[1], z, m[1]);
    Mux_2to1 mx4(v[0], voa[0], z, m[0]);
endmodule
```



Name	Value	Data T...
v[3:0]	f	Array
z	1	Logic
k[31:0]	16	Array
m_out[3:0]	5	Array

```
Name        Value      0 ns      20 ns     40 ns     60 ns     80 ns     100 ns    120 ns    140 ns    160 ns
v[3:0]      f          0         1    2    3    4    5    6    7    8    9    a    b    c    d    e    f
z           1
k[31:0]     0000001    0000... 0000... 0000... 0000... 0000... 0000... 0000... 0000... 0000... 0000... 0000... 0000... 0000... 0000... 0000... 00000010
m_o...:0]   5          0         1    2    3    4    5    6    7    8    9    0    1    2    3    4    5
```

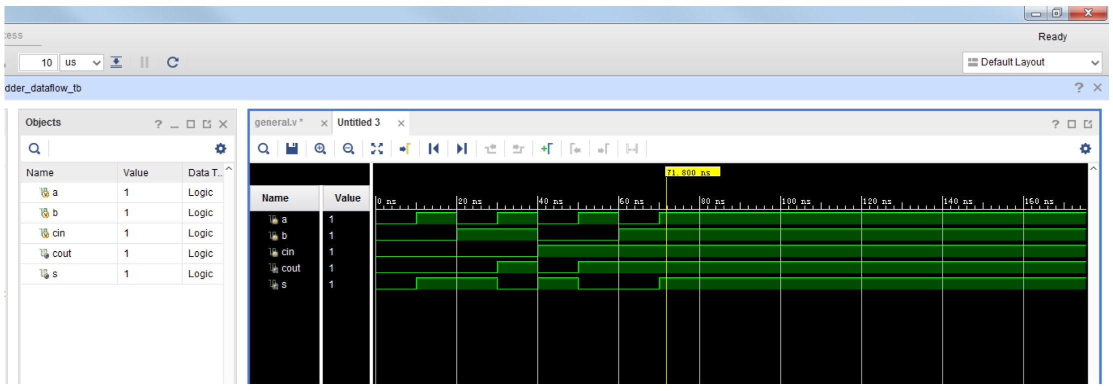
实验 2-2-1 仿真

实验 2-2-1 下载（10~15 情形）

实验 2-2-1 下载（0~9 情形）

//实验 2-3-1 源代码

```
`timescale 1ns / 1ps
module fulladder_dataflow(
    input a,
    input b,
    input cin,
    output s,
    output cout
    );
    assign {cout, s} = a + b + cin;

endmodule

module fulladder_4bit(
    input [3:0] a,
    input [3:0] b,
    input cin,
    output [3:0] s,
    output cout
    );
    wire [2:0] tmp;
    fulladder_dataflow FA1(a[0], b[0], cin, s[0], tmp[0]);
    fulladder_dataflow FA2(a[1], b[1], tmp[0], s[1], tmp[1]);
    fulladder_dataflow FA3(a[2], b[2], tmp[1], s[2], tmp[2]);
    fulladder_dataflow FA4(a[3], b[3], tmp[2], s[3], cout);
endmodule
```



Name	Value	Data T..
a	1	Logic
b	1	Logic
cin	1	Logic
cout	1	Logic
s	1	Logic

实验 2-3-1 仿真截图（一位全加器）

实验 2-3-1 验证（学号后两位 3+2+进位的 1=6）

实验总结：

本次实验我掌握了多模块开发的方法，同时进一步锻炼了仿真功能。实验犯下的一个小错误：数据流写法译码最小项的时候用了 a & b + c & d，而不是(a & b) |(c & d) 【前者会被解释成 a & (b + c) & d，且+为二进制加法，反映到 RTL 是 RTL_ADD，而不是 RTL_AND】，幸亏在 Simulation 的时候看出来了。由此，我也认识到 Testbench 的重要性。