浙江大学实验报告

课程名称：数字系统　　任课老师：李宇波

实验名称：ALU 的仿真　　实验日期：2023/4/21

一、实验目的和要求

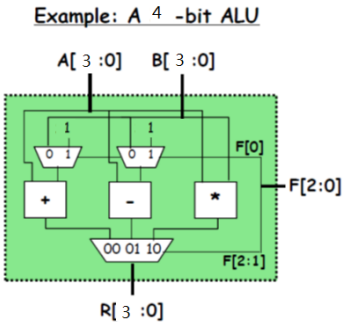
1.1 实验目的

To learn how to write Verilog testbench to simulate the ALU designed in Lab 3.

学习编写 Verilog testbench，仿真设计的 ALU。

1.2 实验要求

Simulate the ALU with the following functions.



Function Table

F2	F1	F0	Function
0	0	0	A + B
0	0	1	A + 1
0	1	0	A - B
0	1	1	A - 1
1	0	X	A * B

1) Simulate the 3-1 mux module by writing a 3-1 mux testbench, then check whether the output is right.

仿真 3-1 mux 模块，检查输出准确性。

2) Simulate the subtract module by writing a subtract testbench, then check whether the output is right.

仿真减法器模块，检查输出准确性。

3) Simulate the ALU module by writing an ALU testbench, then check whether all the function is worked.

仿真整个 ALU，并检查所有函数是否正确工作。

二、实验代码和结果

2.1 仿真 3-1 mux

创建 tb_mux31.v 文件，用于仿真代码的编写。

具体代码：

```
//~ `New testbench
`timescale  1ns / 1ps

module tb_mux31;

// mux31 Parameters
parameter PERIOD  = 10;

reg clk;
reg rst_n;

// mux31 Inputs
reg   [7:0]  d0                         = 0 ;
reg   [7:0]  d1                         = 0 ;
reg   [7:0]  d2                         = 0 ;
reg   [1:0]  s                          = 0 ;

// mux31 Outputs
wire  [7:0]  y                          ;


initial
begin
    forever #(PERIOD/2)  clk=~clk;
end

initial
begin
    #(PERIOD*2) rst_n  =  1;
end

mux31  u_mux31 (
    .d0                     ( d0  [7:0] ),
    .d1                     ( d1  [7:0] ),
    .d2                     ( d2  [7:0] ),
    .s                      ( s   [1:0] ),

    .y                      ( y   [7:0] )
);

initial
begin
    d0 = 8'b01100101;
    d1 = 8'b00100010;
    d2 = 8'b10000011;
```

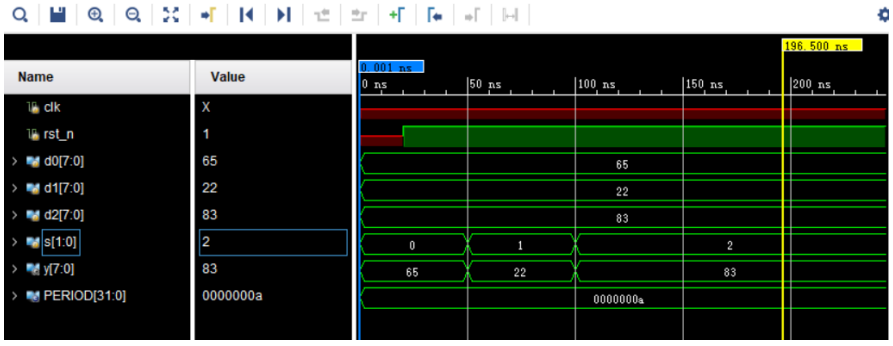
```
    s = 2'b00;
    #(PERIOD*5)
    s = 2'b01;
    #(PERIOD*5)
    s = 2'b10;

end

endmodule
```

简洁起见，下文仿真代码将只展现数据流变化相关的核心内容。

得到的仿真结果如下，



可见，随着 s 的取值改变，3-1 mux 依次输出了 d0，d1，d2 的值。因此仿真结果显示，3-1 mux 的功能完备。

2.2 仿真减法器

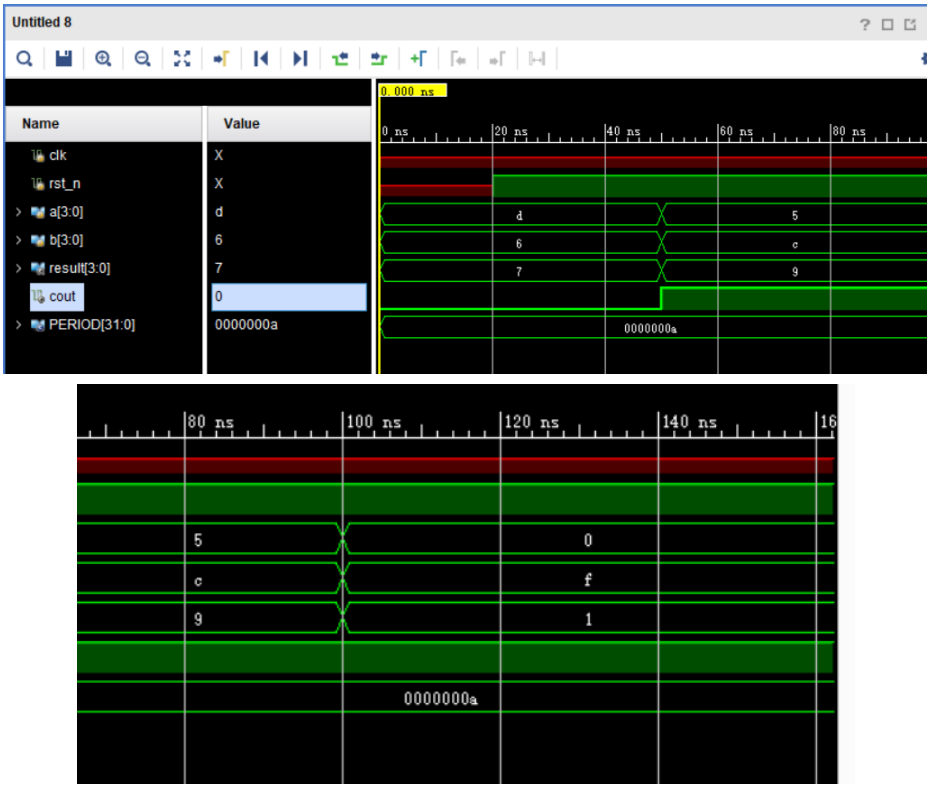
创建 tb_subtractor.v 文件，用以编写仿真代码。

相关代码如下：

```
initial
begin
//  1. needless to borrow digit
    a = 4'b1101; //13
    b = 4'b0110; //6
    #(PERIOD*5)
//  2. need to borrow digit
    a = 4'b0101; //5
    b = 4'b1100; //12
    #(PERIOD*5)
//  4-bit overflow
    a = 4'b0000;
    b = 4'b1111;
end
```

a 为被减数，b 为减数，result 是 4 进制结果，而 cout 为符号信号，0 表示结果为正，1 表示结果为负。

所得仿真结果：



第一段为无借位，a 显示为十六进制的 d，十进制值即 13，b 为 6，result 输出 7，准确；cout 输出 0，准确。

第二段为有借位，b 显示为十六进制的 c，十进制值即 12，a 为 5，result 输出 1001，补码得 0111，即十进制 7，符合二进制减法方法；cout 输出 1，准确。

第三段为考虑溢出的借位，a 是 0，b 是 15，result 输出 0001，补码得 1111，准确；cout 输出 1，准确。

因此认为减法器功能完备。

2.3 仿真 ALU

创建 tb_ALU.v 文件，用以编写仿真代码。

相关代码如下：

```
initial
begin
//  1. A + B
    F = 3'b000;
    A = 4'b0011; // A = 3
    B = 4'b1011; // B = 11
    #(PERIOD*2)

//  2. A + B with overflow
    B = 4'b1110; // B = 14
    #(PERIOD*2)

//  3. A + 1
```

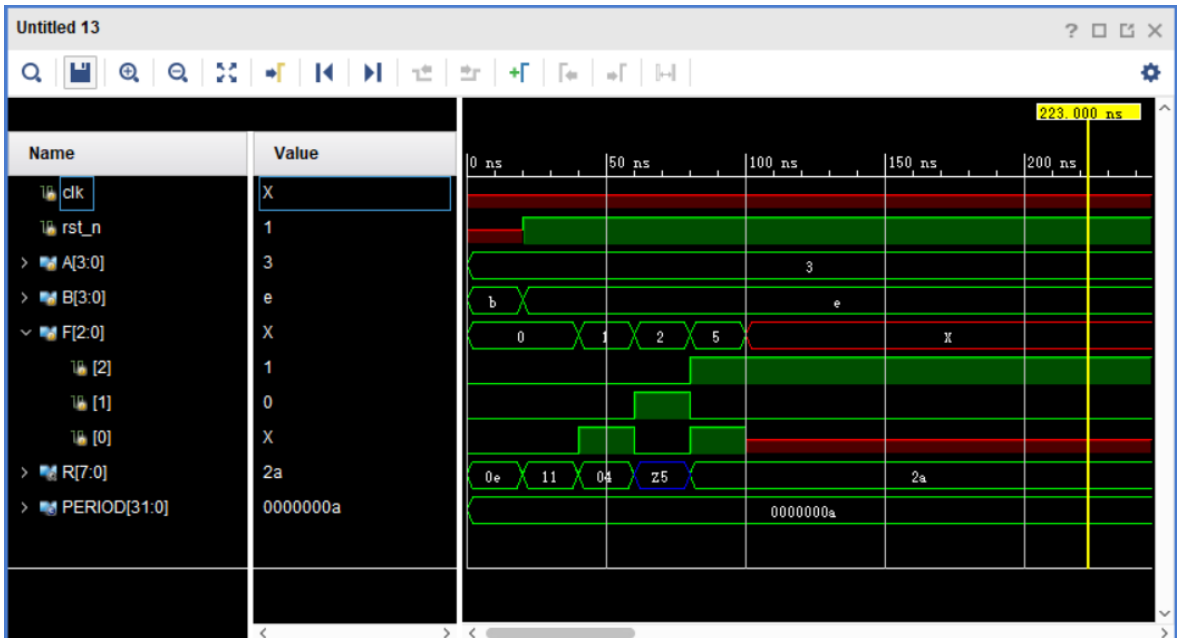
```
    F = 3'b001;
    #(PERIOD*2)

//  4. A - B
    F = 3'b010;
    #(PERIOD*2)

//  5. A - 1
    F = 3'b011;
    #(PERIOD*2)

//  6. A * B
    F = 3'b10x;
end
```

所得仿真结果：

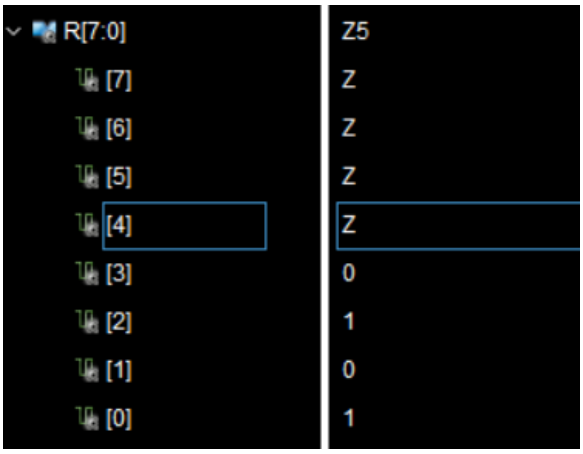


测试一，令 A=3，B=11，仿真 A+B，结果 R 为 0e，即十进制 14，准确。

测试二，令 A=3，B=14，仿真 A+B，结果 R 为 10001(由 cout 和 result 拼接，cout 是 1bit，result 是 4bit，因此 R 展开应为 10001)，即十进制 17，准确。

测试三，仿真 A+1，结果 R 为 04，即十进制 4，准确。

测试四，仿真 A-B，将 R 展开，0101，补码后为 1011，即十进制 11，准确。



测试五，仿真 A-1，R 为 2，准确。

测试六，仿真 A*B，R 为十六进制 2a，即十进制 42，准确。

因此认为 ALU 功能完备。