

实例学习 Robei 芯片设计系列

四. ALU 设计

Robei LLC

1. 实验目的

ALU（算数逻辑单元）是 CPU 的基本组成部分。实验要求掌握算术逻辑运算加、减操作原理，验证运算器的组合功能。

2. 实验准备

ALU 的基本结构如图 1 所示。我们所设计的 ALU 要实现最基本的加减运算，与或非和异或等功能。

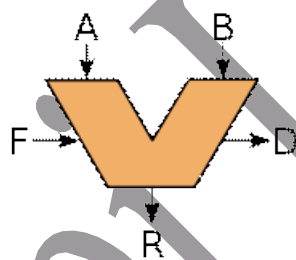


图 1. ALU 基本结构

(1) 加法运算包含 2 种类型，一种是不带进位的加法器，另外一种带进位的加法器。不带进位的加法器的公式：

$$\{D, R\} = A + B \tag{1}$$

带进位的可以进行加法器级联，实现更高位数的串行加法运算。带进位的加法器的公式：

$$\{D, R\} = A + B + F \tag{2}$$

(2) 减法运算也包含 2 种类型。不带借位的减法运算：

$$\{D, R\} = A - B \tag{3}$$

带借位的减法运算：

$$\{D, R\} = A - B - F \tag{4}$$

3. 实验要求

设计一个 8 位 ALU，并能实现数据与，或，非，异或，不带进位加法，带进位加法，不带借位减法和带借位减法运算。运算符采用 3 比特表示。A, B, R 均为 8 比特数据。用测试文件测试你的 ALU 功能，并用级联方式将 4 个 8 比特的 ALU 实现 32 比特的 ALU。

4. 实验内容

4.1 ALU 模型设计

- 1) 新建一个模型命名为 alu，类型为 module，同时具备 4 输入 2 输出。每个引脚的属性和名称参照图 2 进行对应的修改。

Name	Inout	Data Type	Datasize	Function
A	input	wire	7:0	first input
B	input	wire	7:0	second input
op	input	wire	3:0	operation
R	output	wire	7:0	result
D	output	wire	1	carry out
F	input	wire	1	Carry in

图 2. 引脚属性

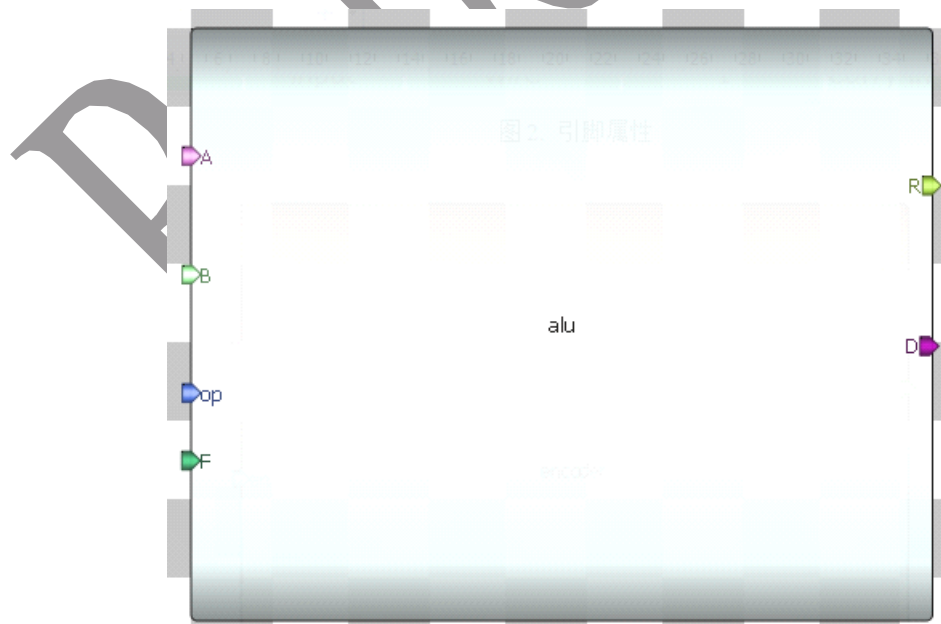
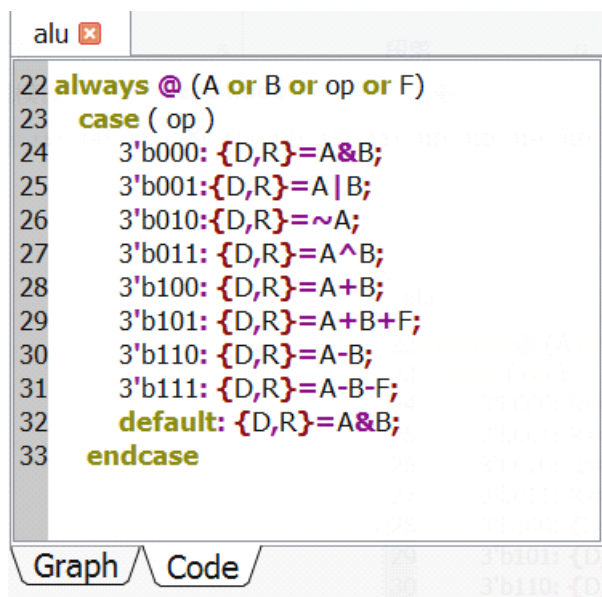


图 3. ALU 界面图

2) 添加代码。点击模型下方的 Code（如图 4 所示）添加代码。



```
22 always @ (A or B or op or F)
23 case ( op )
24     3'b000: {D,R}=A&B;
25     3'b001: {D,R}=A|B;
26     3'b010: {D,R}=~A;
27     3'b011: {D,R}=A^B;
28     3'b100: {D,R}=A+B;
29     3'b101: {D,R}=A+B+F;
30     3'b110: {D,R}=A-B;
31     3'b111: {D,R}=A-B-F;
32     default: {D,R}=A&B;
33 endcase
```

图 4. 点击 Code 输入算法

在代码设计区内输入以下 Verilog 代码：

```
always @ (A or B or op or F)
case ( op )
    3'b000: {D,R}=A&B; //实现与运算
    3'b001: {D,R}=A|B; //实现或运算
    3'b010: {D,R}=~A; //实现非运算
    3'b011: {D,R}=A^B; //实现异或运算
    3'b100: {D,R}=A+B; //实现不带进位的加运算
    3'b101: {D,R}=A+B+F; //实现带进位的加运算
    3'b110: {D,R}=A-B; //实现不带借位的减运算
    3'b111: {D,R}=A-B-F; //实现带借位的减运算
    default: {D,R}=A&B; // 默认为与运算
endcase
```

3) 保存模型到一个文件夹中，运行并检查有无错误输出。

4.2 测试文件设计

1) 新建一个 4 输入 2 输出的测试文件，记得将 Module Type 设置为“testbench”各个引脚配置如图 5 所示。

Name	Inout	DataType	Datasize	Function
a	input	reg	7:0	first input
b	input	reg	7:0	second input
op	input	reg	3:0	operation
cin	input	wire	1	carry in
result	output	wire	7:0	result
cout	output	wire	1	carry out

图 5. 新建测试文件

- 2) 另存为测试文件。将测试文件保存到 alu 模型所在的文件夹下。
- 3) 加入模型。在 Toolbox 工具箱的 Current 栏里，会出现一个 alu 模型，单击该模型并在 alutest 上添加，并连接引脚。

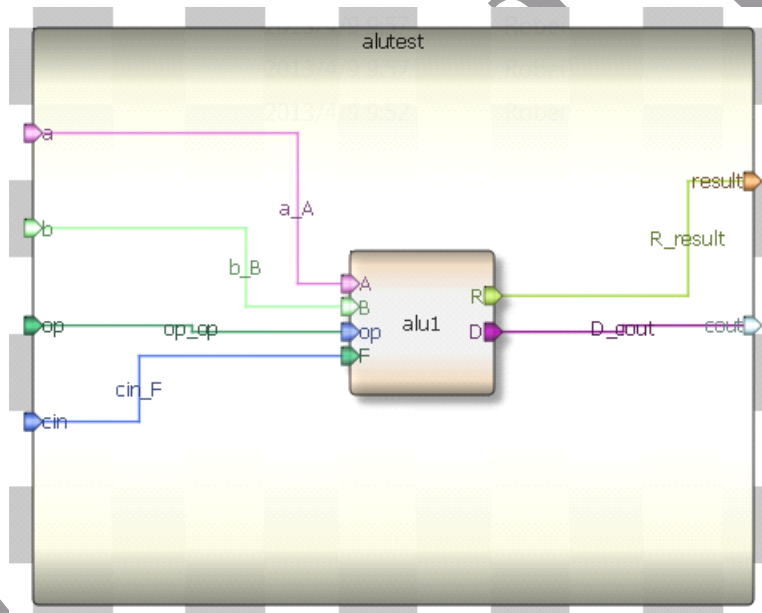


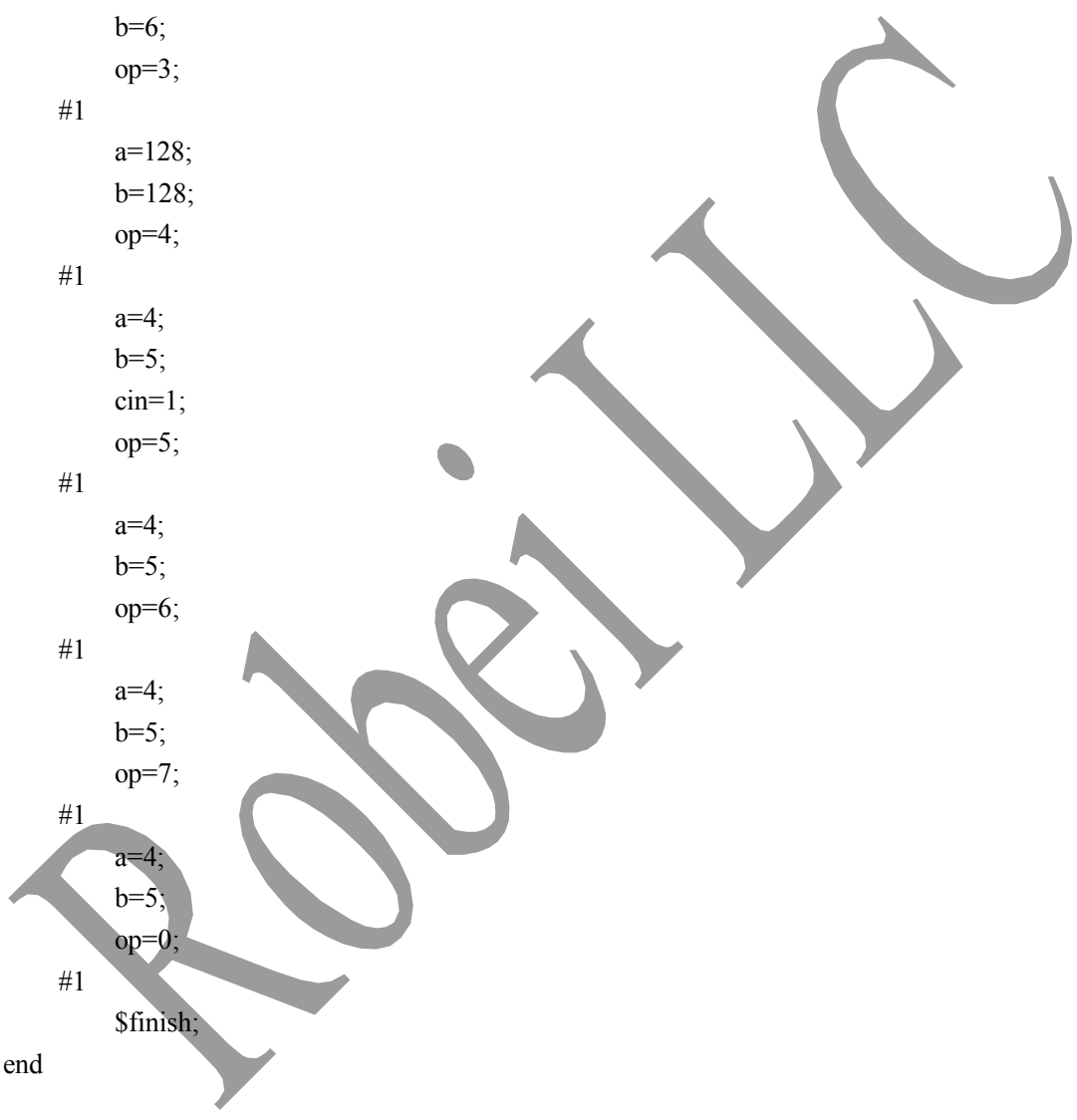
图 6. 添加模型

- 4) 输入激励。点击测试模块下方的“Code”，输入激励算法。激励代码在结束的时候要用\$finish 结束。

```

initial begin
    a=0;
    b=0;
    op=0;
    cin=0;
    #1
    a=3;
    b=1;
    op=0;
    #1
    
```

```
a=2;
b=1;
op=1;
#1
a=255;
b=0;
op=2;
#1
a=5;
b=6;
op=3;
#1
a=128;
b=128;
op=4;
#1
a=4;
b=5;
cin=1;
op=5;
#1
a=4;
b=5;
op=6;
#1
a=4;
b=5;
op=7;
#1
a=4;
b=5;
op=0;
#1
$finish;
end
```



```
alutest x
15 initial begin
16   a=0;
17   b=0;
18   op=0;
19   cin=0;
20 #1
21   a=3;
22   b=1;
23   op=0;
24 #1
25   a=2;
26   b=1;
27   op=1;
28 #1
29   a=255;
30   b=0;
31   op=2;
32 #1
33   a=5;
34   b=6;
35   op=3;
36 #1
```

图 7 激励代码

- 5) 执行仿真并查看波形。查看输出信息。检查没有错误之后查看波形。点击右侧 Workspace 中的信号，进行添加并查看分析仿真结果。对照真值表，查看设计波形输入输出是否一致。

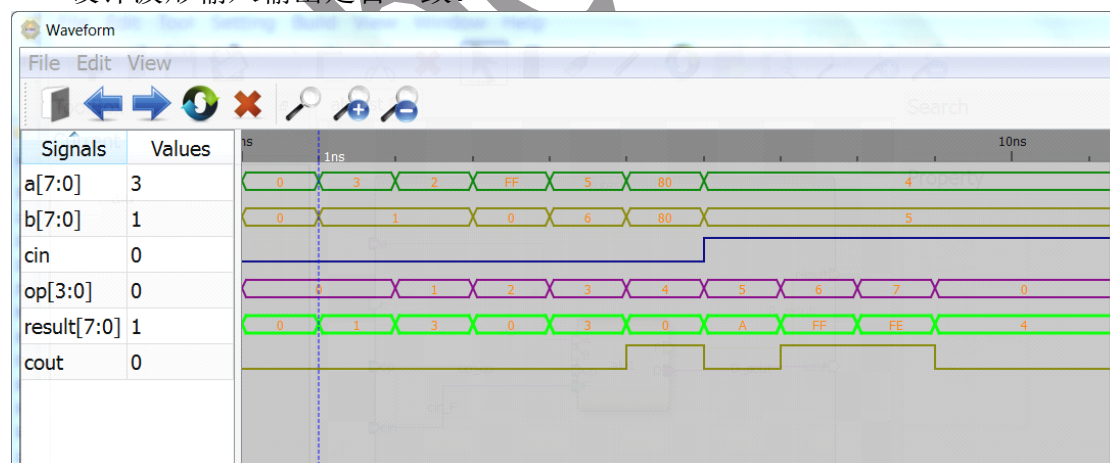


图 8. 查看波形

4.3 32 位 ALU 设计

- 1) 我们利用 8 位的 ALU 级联来设计一个 32 位的 ALU，这个实验需要先行注册 Robei 软件，否则不能进行仿真。

- 2) 创建一个新的模型，添加 10 个输入引脚，5 个输出引脚，各个引脚的配置如图 9 所示。保存到 alu 模型所在的文件夹。

Name	Inout	DataType	Datasize	Function
A0	input	wire	7:0	Bit 0~7 of A
B0	input	wire	7:0	Bit 0~7 of B
A1	input	wire	7:0	Bit 8~15 of A
B1	input	wire	7:0	Bit 8~15 of B
A2	input	wire	7:0	Bit 16~23 of A
B2	input	wire	7:0	Bit 16~23 of B
A3	input	wire	7:0	Bit 24~31 of A
B3	input	wire	7:0	Bit 24~31 of B
op	input	wire	3:0	operation
R0	output	wire	7:0	Bit 0~7 of R
R1	output	wire	7:0	Bit 8~15 of R
R2	output	wire	7:0	Bit 16~23 of R
R3	output	wire	7:0	Bit 24~31 of R
D	output	wire	1	carry out
F	input	wire	1	carry in

图 9. 32 位 ALU 引脚

- 3) 添加 4 个 ALU 连接引脚。如图 10 所示。4 个 8 位的 ALU 进行级联，第一个输出的 D 连到下一级的 F，最终的 ALU 的 D 连接到顶层的 D 引脚。第一个 ALU 的 F 连接到顶层模块的 F。op 都连接到顶层的 op 引脚上，A、B 和 R 按照高低位进行连接。这样输入 $A_3A_2A_1A_0$ 、 $B_3B_2B_1B_0$ 和 $R_3R_2R_1R_0$ 分别是 32 位 ALU 的输入和输出端。

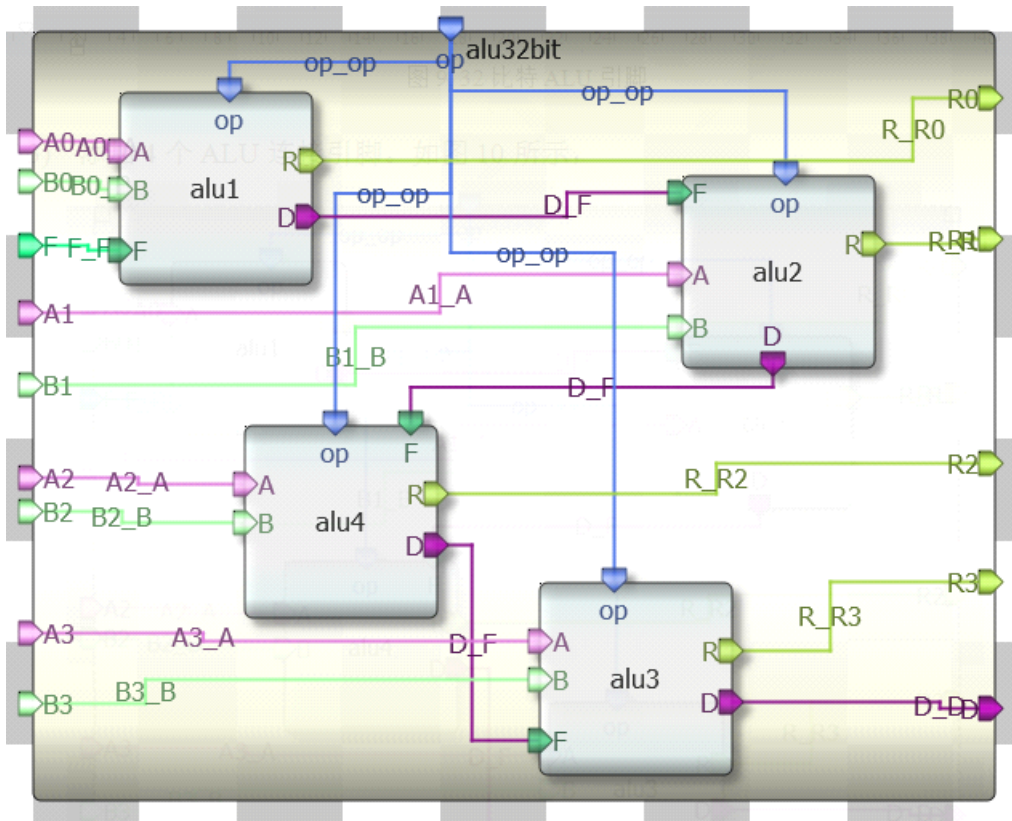


图 10. 32 位 ALU 设计图

- 4) 创建一个测试文件，10 个输入引脚 5 个输出，按照图 11 进行引脚配置并保存到与 alu32bit 模型同一个文件下。

Name	Inout	DataType	Datasize	Function
A0	input	reg	7:0	Bit 0~7 of A
B0	input	reg	7:0	Bit 0~7 of B
A1	input	reg	7:0	Bit 8~15 of A
B1	input	reg	7:0	Bit 8~15 of B
A2	input	reg	7:0	Bit 16~23 of A
B2	input	reg	7:0	Bit 16~23 of B
A3	input	reg	7:0	Bit 24~31 of A
B3	input	reg	7:0	Bit 24~31 of B
op	input	reg	3:0	operation
R0	output	wire	7:0	Bit 0~7 of R
R1	output	wire	7:0	Bit 8~15 of R
R2	output	wire	7:0	Bit 16~23 of R
R3	output	wire	7:0	Bit 24~31 of R
D	output	wire	1	carry out
F	input	reg	1	carry in

图 11. 32 位 ALU 测试文件引脚配置

- 5) 从 Toolbox 里面的 Current 栏找 alu32bit 模型，并添加到测试模块上。对应引脚相连。如图 12 所示。

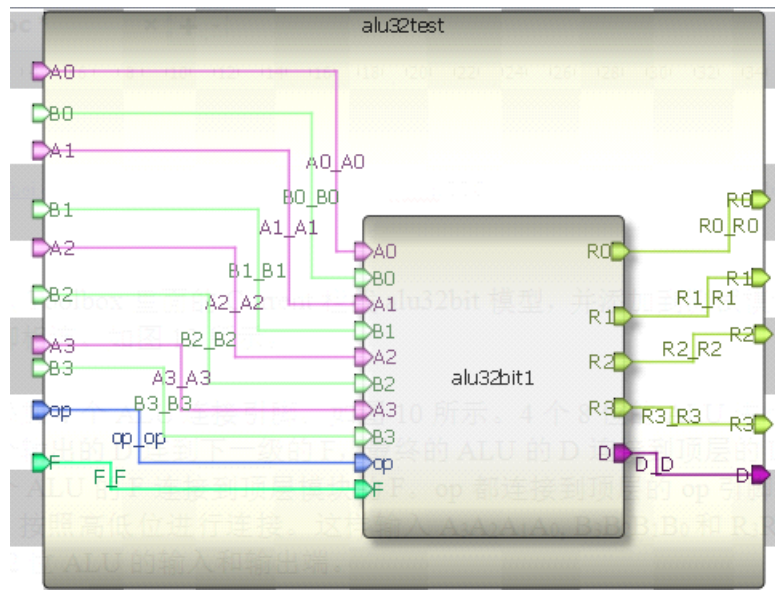


图 12. 32 位 ALU 测试引脚连接

- 6) 自己设计测试激励代码，并仿真查看结果。

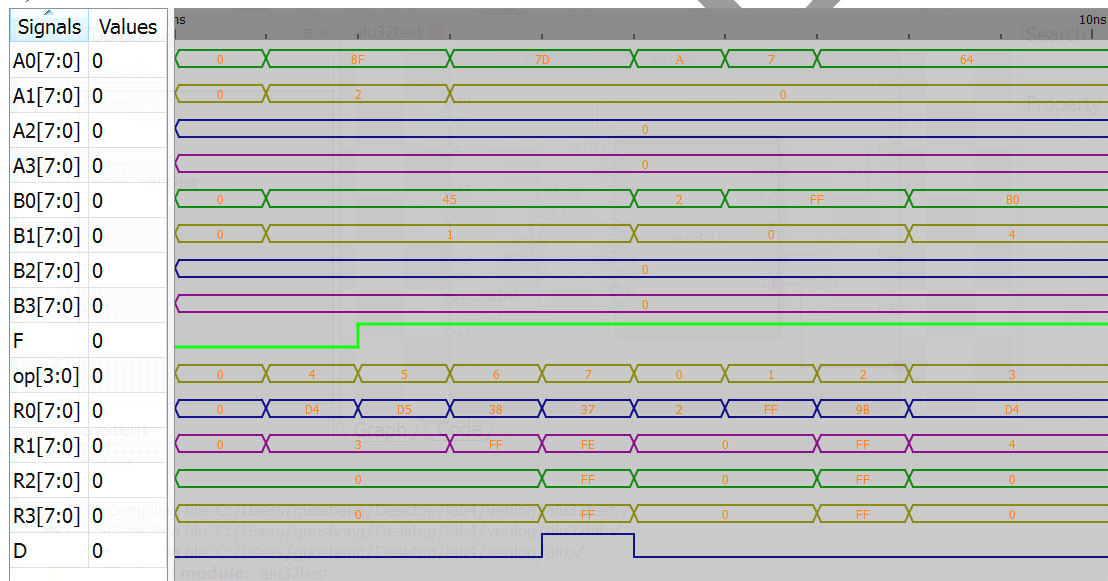


图 13. 32 位 ALU 仿真波形

5. 问题与思考

- 1) 不要使用 8 位 ALU 级联的方式，直接用 Verilog 在 Robei 中实现一个 32 位或者 64 位 ALU。
- 2) **挑战题：**在 8 位 ALU 设计上添加乘法功能，输出结果变成 16 位输出。利用这个 ALU 实现一个 16 位的乘法器。提示：16 位乘法器分成低 8 位和高 8

位。如 $A[15:0]$ 拆分成 $A[15:8]$ 和 $A[7:0]$ ，同样拆分 B 。之后用 4 个乘法器分别实现：

- $A[7:0] \times B[7:0]$
- $A[7:0] \times B[15:8]$
- $A[15:8] \times B[7:0]$
- $A[15:8] \times B[15:8]$

然后进行适当移位，再用加法器实现相加。

Robei LLC