Step By Step Learning Robei

IV. ALU design

Robei LLC

1. Objective

ALU (Arithmetic Logic Unit) is the fundamental element in central process unit in computer, which performs integer arithmetic and logic operations. In this lab, we will design a simple ALU unit to get familiar with arithmetic operation and logic operation.

2. Preparation

The structure of ALU shows in fig. 1. We want to design an ALU that can perform addition, subtraction, AND, OR and XOR operation.

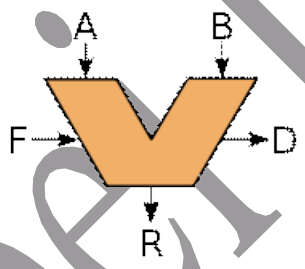

Fig.1. ALU structure

(1) Addition has two different types, one is addition without carry, another is addition with carry. The difference is depending on whether carry in has been used in calculation. The equation of addition without carry in shows in Equation 1.

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

Addition with carry in can realize more bits of calculation by cascading many of them. he equation of addition with carry in shows in Equation 2.

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

(2) Substraction also including two types: subtraction without borrow, and subtraction with borrow. The first type of subtraction show in Equation 3 and the second type shows in Equation 4.

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

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

3. Requirement

Design an eight bits ALU with functions of AND, OR, NOT, XOR, addition without carry in, addition with carry in, subtraction without borrow and subtraction with borrow. Operation use 3 bits to control, ports A, B, R are 8 bits. After this design, verify your ALU with testbench. Design a 32 bit ALU with 4 eight bits ALU cascading.

4. Procedure

4.1 8 bits ALU Design

1) Create a new module with name "alu" with 4 input ports and 2 output ports. Modify the corresponding ports with the following properties in fig. 2.

Name	Inout	DataType	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

Fig. 2. Ports' properties

Fig. 3. ALU interface

2) Add algorithm code. Click "Code" tab under design, and type in the following code.

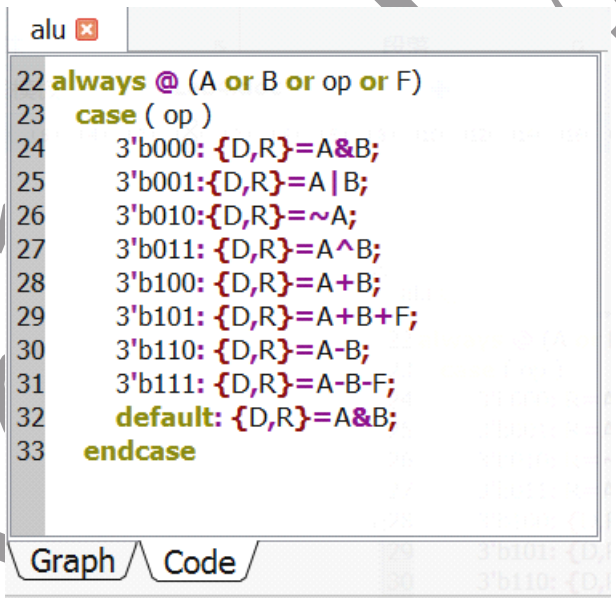

Fig. 4. Add code algorithm

```
always @ (A or B or op or F)
    case ( op )
        3'b000: {D,R}=A&B;    // AND operation
        3'b001: {D,R}=A|B;    // OR operation
        3'b010: {D,R}=~A;     // NOT operation
        3'b011: {D,R}=A^B;    // XOR operation
        3'b100: {D,R}=A+B;    //Add without carry
        3'b101: {D,R}=A+B+F; //Add with carry
        3'b110: {D,R}=A-B;    //Sub without borrow
```

```
        3'b111: {D,R}=A-B-F;    //Sub with borrow
        default: {D,R}=A&B;    // Default is AND operation
    endcase
```

3) Save module to a folder, and check whether there are any errors in output window.

4.2 Testbench design

1) Create a new Testbench with 4 input ports and 2 output ports. Set the Module Type to “testbench”. Following the properties in Fig.5 to modify corresponding ports.

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

Fig. 5. Testbench Ports

2) Save as testbench. save testbench to the same location as “alu”.
3) Add model. Under category “Current” in Toolbox, there will be one model showing up. Add the “alu” model on testbench design, and connect ports like fig. 6.

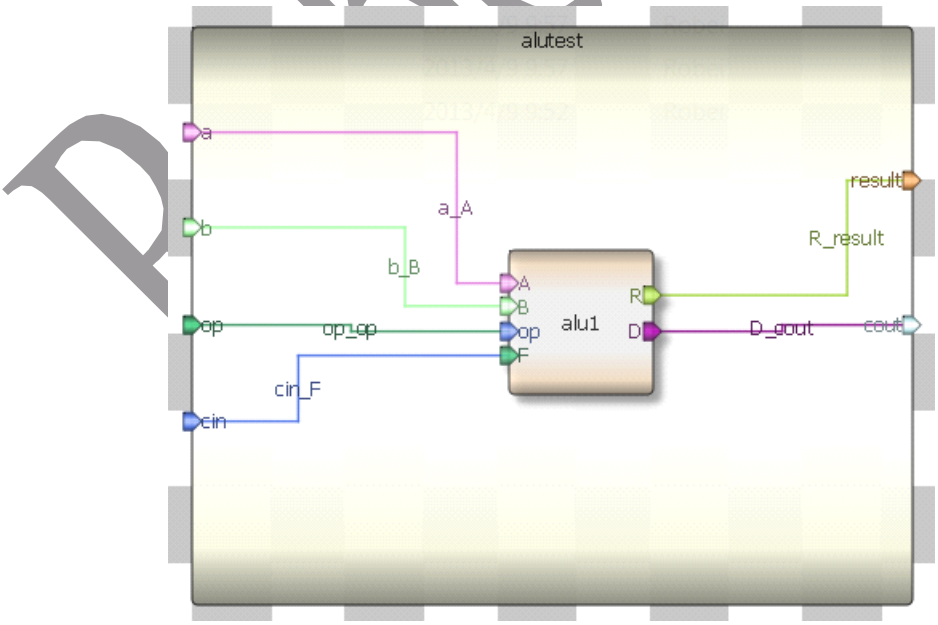

Fig. 6. Add model

4) Code stimulate. Press “Code” tab under testbech, type in the following code as stimulate for simulation. Remember to use $finish to end this simulation.

```
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

```
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
```

Graph Code

Fig. 7 Stimulate code

5) Run simulation and check the waveform. Click on signals listed in Workspace, add to wave window. Analyze the waveform result and compare with your lab requirement.

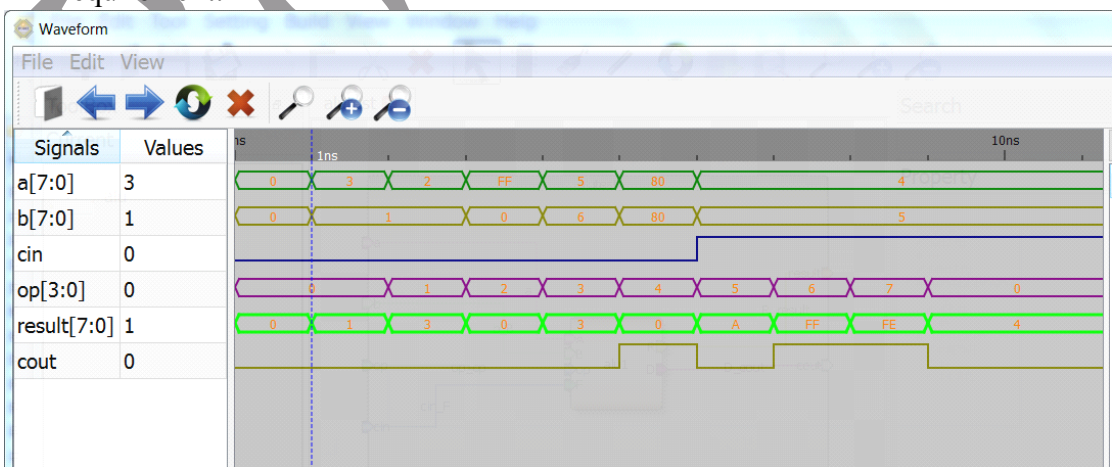

Fig. 8. Check waveform

4.3 32 bits ALU design

1) Before you start this lab, please make sure your Robei software registered, otherwise, you can not simulate with Robei. We will cascade 8 bits ALUs to form a 32 bits ALU.
2) Create a new module "alu32bit" with 10 input ports and 5 output ports. The properties of each port shows in Fig. 9. Save file to the save location as "alu".

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

Fig. 9. 32 bits ALU

3) Add 4 ALUs as fig. 1o, and cascade them. Inputs $A_3A_2A_1A_0$, $B_3B_2B_1B_0$ 和 $R_3R_2R_1R_0$ are 32 bits ALU's input and output.

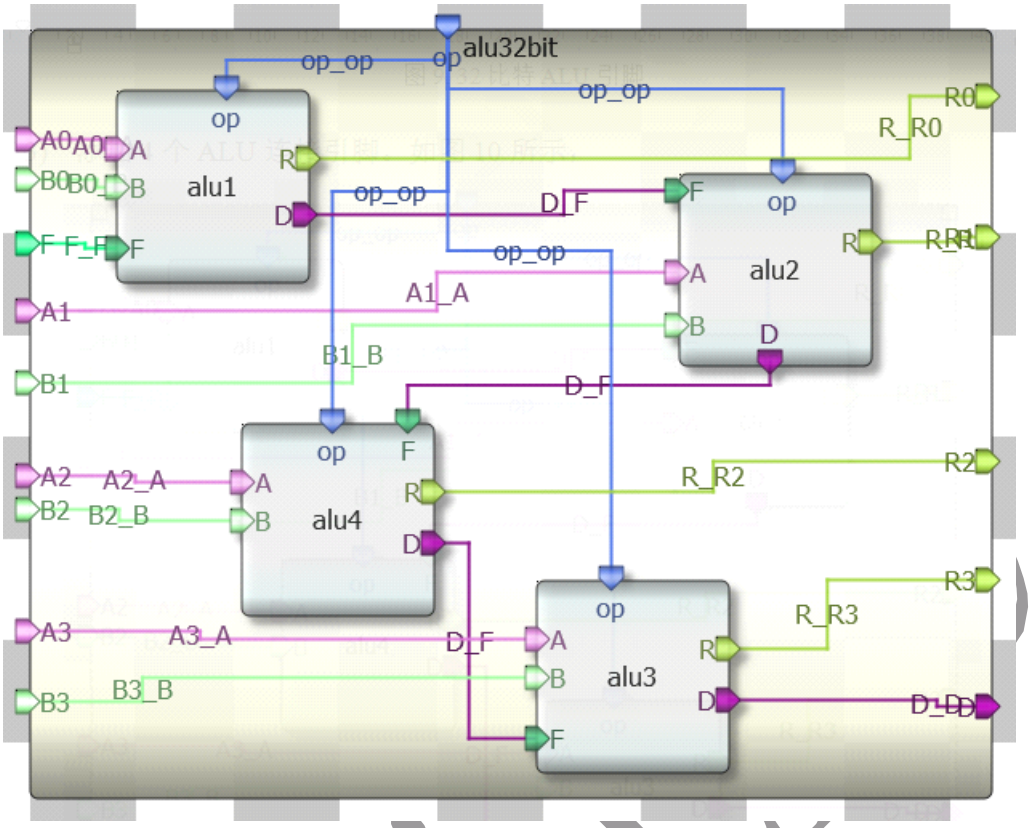

Fig. 10. 32 bits ALU connections

4) Crete a testbench with 10 input ports and 5 output ports, following fig. 11 to configure each ports. Save the testbench to same location as "alu32bit" model.

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

Fi.g 11. 32 bits ALU testbench ports

5) Add “alu32bit” model from “Current” category in Toolbox to testbench. Connect corresponding ports as fig. 12.

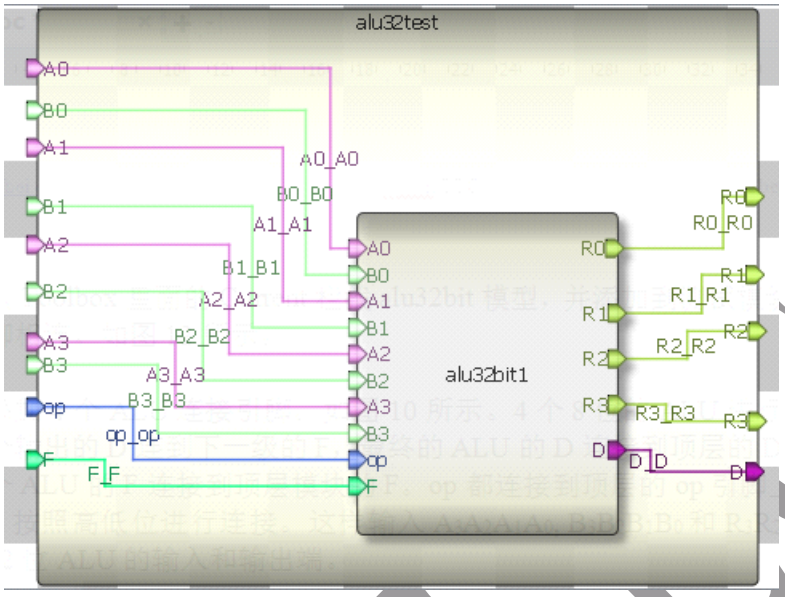

Fig. 12. 32 bit ALU testbench connection

6) Code your own testbench code, and check the simulation result in waveform.

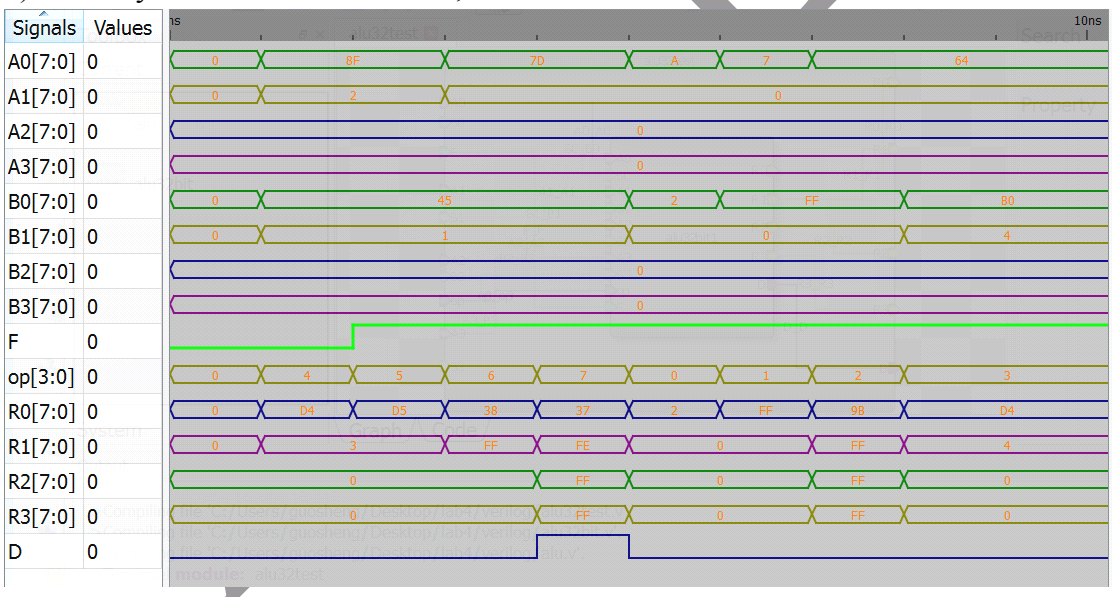

Fig. 13. 32 bits ALU waveform

5. Questions

1) Without using 8 bits ALU cascading, use Verilog in Robei to design a 32 or 64 bits ALU.
2) Challenge：Add multiplier in 8 bits ALU, let the output has 16 bits. Use this ALU to realize a 16 bits multiplier.

Note: 16 bits multiplier can be divided into high 8 bits and low 8 bits, like A[15:0] can be divided into A[15:8] and A[7:0]. Use four multiplier to realize：

- A[7:0]×B[7:0]
- A[7:0]×B[15:8]
- A[15:8]×B[7:0]
- A[15:8]×B[15:8]

Then shift corresponding bits and add them together.