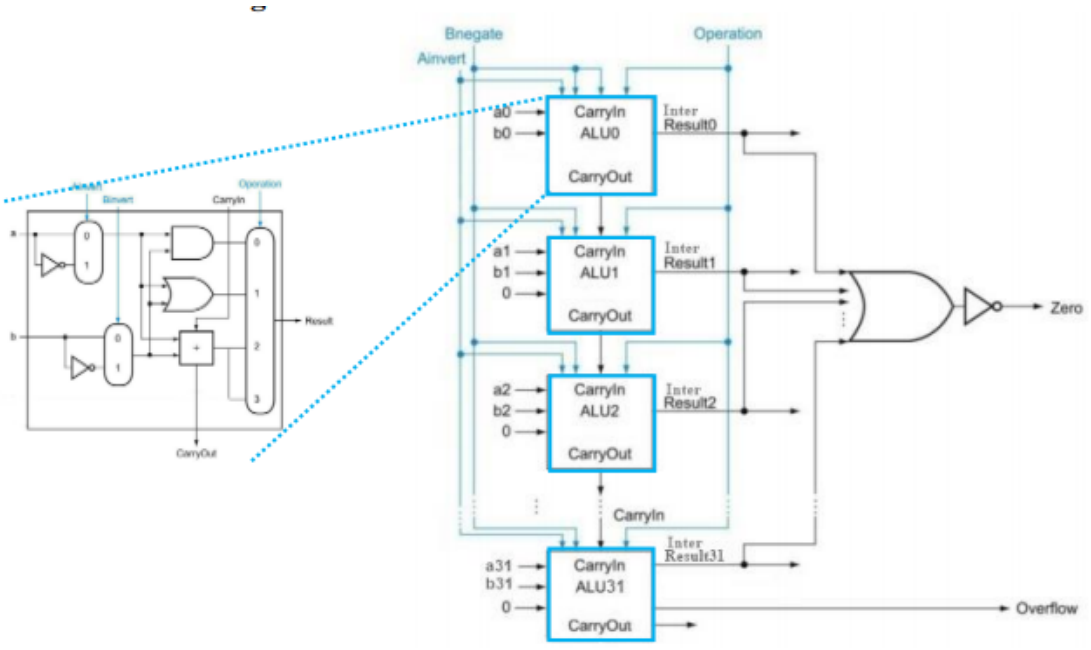
Computer Organization lab 2

Information

- Author : 0712238, Yan-Tong Lin, 林彥彤
- Usage : Computer Organization Lab 2 for TF cheng's class
- Due date : 2020/4/20
- Version control : https://github.com/EazyReal/Computer-Organization-2020 (https://github.com/EazyReal/Computer-Organization-2020)
- Online version of this MD file
  - Computer Organization lab 2 (https://hackmd.io/8HEfOozyQau8GKq4SUOxQw)

Architecture design and diagram



- ALU_top is same as the diagram
- Sequencial Architecture is adopted as the diagram
- the result[0:31] is renamed as $result_t$[0:31]
- which is mux with 32-bit 1/0 determined by lt(less than)(1 bit wire)
  - lt is a function of src1[31], src2[31], carry[31]
  - lt is designed with case by case study

```
assign neq31 = src1[31] ^ ~src2[31]; //a, -b same sign or not
//not same => see result(no overflow possible when adding a, -b)
//same => use a's sign(no need to see substracy res to know)
assign lt = neq31 ? ~carry[31] : src1[31];
//assign res = is_slt ? (lt ? 1 : 0) : res;
assign result = (ALU_control == 4'b0111) ? (lt ? 32'h01 : 32'h00 ) : result_t;
```

- the mux outcome(determined by ALU_Control) is the new result[0:31]
- then is the desired result and is fed to get zero flag

Detailed description of the implementation

- use **ALU_Control** to set **a_in(a = ~a), b_in, opcode** to achieve desired calculations

- **And**
  - res = a&b
  - cout = 0
- **Or**
  - res = a|b
  - cout = 0
- **Nor**
  - a = ~a, b = ~b
  - res = a&b
  - cout = 0
- **Nand**
  - a = ~a, b = ~b
  - res = a|b
  - cout = 0
- **Add**
  - res = a xor b xor cin
  - cout = $s_1s_2+s_1cin+s_2cin$
- **Sub**
  - b = ~b+1, achived by cin[0]=1 in control
  - res = a xor b xor cin
  - cout = $s_1s_2+s_1cin+s_2cin$
- **SLT**
  - set the alu direct result as wire **result_t**
  - final result(output wire) is set to 32bit 1 or 0 determined by lt(wire) when ALU_control is slt(4'b0111)
  - lt calculation:
    - neq31=src1[31] src2[31]

        - check whether $a, -b$ is of same sign
    - lt=neq31? carry[31]:src1[31]
        - if is of same sign => use one of them as result
        - else => no overflow, use carry in of bit 31 to determine if $a-b$ is regetive

- **Flags**:
    - Zero
        - not (or value of 31 bits esult)
    - Carry
        - carry out of 31^{th} bit
    - Overflow
        - when carry in$\neq$carry out in last bit
        - overflow=carry[31]$\oplus$cout

Implementation results

ALU.v

```
/************************************************
Student Name: Yan-Tong Lin
Student ID: 0712238
Date: 2020/04/20
************************************************/

`timescale 1ns/1ps

module alu(
    rst_n,         // negative reset            (input)
    src1,          // 32 bits source 1          (input)
    src2,          // 32 bits source 2          (input)
    ALU_control,   // 4 bits ALU control input  (input)
    result,        // 32 bits result            (output)
    zero,          // 1 bit when the output is 0, zero must be set (output)
    cout,          // 1 bit carry out           (output)
    overflow       // 1 bit overflow            (output)
);


input           rst_n;
input  [32-1:0] src1;
input  [32-1:0] src2;
input  [4-1:0]  ALU_control;

output [32-1:0] result; //output is a wire
output          zero;
output          cout;
output          overflow;

//tmp result to mux with slt
wire [32-1:0]  result_t;

//flag wire
wire            zero;
wire            cout;
wire            overflow;

//for slt
wire            neq31;
wire            lt;

//control
reg [1:0] oper;
wire [31:0] carry; //carry[i] means carry "for" i th bit, not "of" i th bit
reg a_in;
reg b_in;

//2-complement for -b
assign carry[0] = (ALU_control==4'b0110)? 1: (ALU_control==4'b0111)? 1: 0; //sub or s

//ZCY flag
assign zero = (result == 0) ? 1 : 0; //implicit nor all output gate!
assign overflow = carry[31] ^ cout; //
//carry is assigned below

//slt judge and set value
assign neq31 = src1[31] ^ ~src2[31]; //a, -b same sign or not
//not same => see result(no overflow possible when adding a, -b)
//same => use a's sign(no need to see substracy res to know)
assign lt = neq31 ? ~carry[31] : src1[31];
//assign res = is_slt ? (lt ? 1 : 0) : res;
assign result = (ALU_control==4'b0111)? (lt ? 32'b01 : 32'b00 ) : result_t;
```

```
63  assign result = (ALU_control == 4'b0111) ? (lt ? 32'h01 : 32'h00 ) : result_t;
64
65
66  always@(*)begin
67          if(rst_n==1)begin
68
69                  case(ALU_control)
70                          4'b0000:begin//And
71                                                  a_in    <= 0;
72                                                  b_in    <= 0;
73                                                  oper    <= 2'b00;//and
74                                                  end
75                          4'b0001:begin//Or
76                                                  a_in    <= 0;
77                                                  b_in    <= 0;
78                                                  oper    <= 2'b01;//or
79                                                  end
80                          4'b0010:begin//Add
81                                                  a_in    <= 0;
82                                                  b_in    <= 0;
83                                                  oper    <= 2'b10;//add
84                                                  end
85                          4'b0110:begin//Sub
86                                                  a_in    <= 0;
87                                                  b_in    <= 1;
88                                                  oper    <= 2'b10;//add
89                                                  end
90                          4'b1100:begin//Nor
91                                                  a_in    <= 1;
92                                                  b_in    <= 1;
93                                                  oper    <= 2'b00;//and
94                                                  end
95                          4'b1101:begin//Nand
96                                                  a_in    <= 1;
97                                                  b_in    <= 1;
98                                                  oper    <= 2'b01;//or
99                                                  end
100                         4'b0111:begin//SLT(set less than)
101                                                 a_in    <= 0;
102                                                 b_in    <= 1;
103                                                 oper    <= 2'b11;//slt
104                                                 end
105                         default: ;
106                 endcase
107
108         end
109 end
110
111 //for loop declaration of ALU0-30
112 parameter NBIT = 30;
113 generate
114 genvar i;
115 for (i=0; i<=NBIT; i=i+1)
116 begin: aluarray
117 alu_top alui( .src1(src1[i]), .src2(src2[i]), .A_invert(a_in), .B_invert(b_in),
118                                     .cin(carry[i]), .operation(oper), .result(result_t[i
119 end
120 endgenerate
121
122 //31 diff at cout.
123 alu_top alu31( .src1(src1[31]), .src2(src2[31]), .A_invert(a_in), .B_invert(b_in),
124                                         .cin(carry[31]), .operation(oper), .result(resu
125
126
```

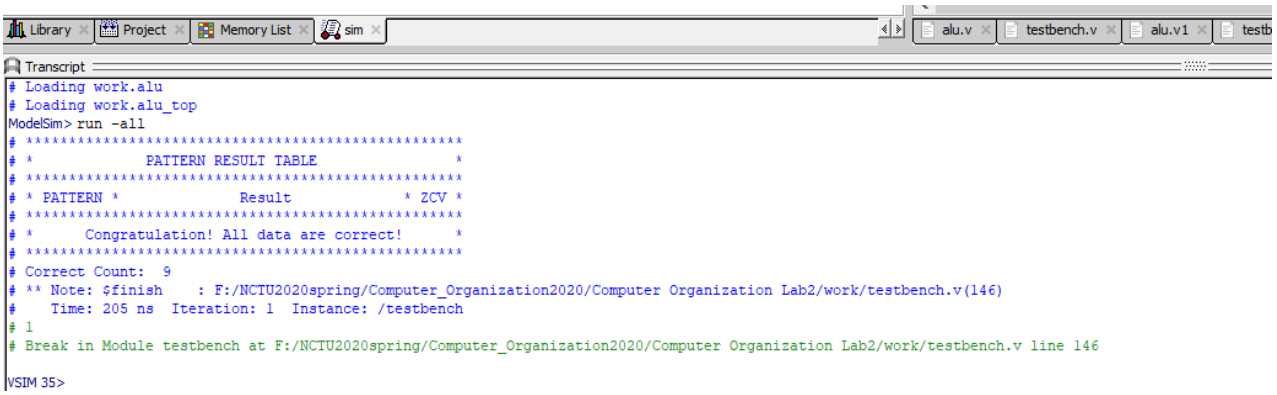
```
127 endmodule
```

ALU_top.v

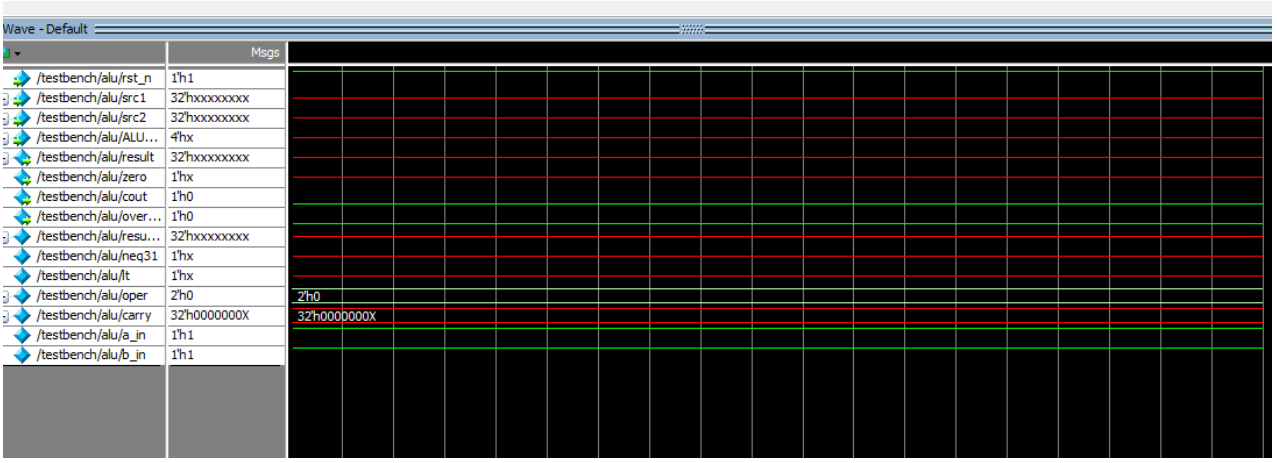
```
/************************************************
Student Name: Yan-Tong Lin
Student ID: 0712238
Date: 2020/04/20
************************************************/

`timescale 1ns/1ps

module alu_top(
        src1,       //1 bit source 1 (input)
        src2,       //1 bit source 2 (input)
        A_invert,   //1 bit A_invert (input)
        B_invert,   //1 bit B_invert (input)
        cin,        //1 bit carry in (input)
        operation,  //operation      (input)
        //set,       //1 bit set     (input) signal from alu.v
        result,     //1 bit result   (output)
        cout       //1 bit carry out(output)
);

input         src1;
input         src2;
input         A_invert;
input         B_invert;
input         cin;
input [2-1:0] operation;
//input         set;

output   reg  result; //can be ressign
output   reg   cout;

reg s1, s2;

always@( * )begin

        s1 = A_invert ? ~src1 : src1;
        s2 = B_invert ? ~src2 : src2;

        case(operation)
                2'b00:begin//And
                        result = s1 & s2;
                        cout = 0;
                        end

                2'b01:begin//Or
                        result = s1 | s2;
                        cout = 0;
                        end

                2'b10:begin//Add
                        result = s1 ^ s2 ^ cin;
                        cout = (s1&s2) + (s1&cin) + (s2&cin);
                        end

                2'b11:begin//SLT
                        result = 0;
                        cout = (s1&s2) + (s1&cin) + (s2&cin);
                        end
        endcase

end

endmodule
```

Execution Result

Answer Correct on testbench.v



Wave



Problems encountered and solutions

- strange for loop bug
  - fixed by checking github version that is not bugged

- still dont know why exactly there was bug

```
//diff at lt, signal if substraction is negative or same sign(a, -b) and sign a is neg tive
alu_top alu0( .src1(src1[0]), .src2(src2[0]), .set(lt), .A_invert(a_in), .B_invert(b_in),
                                  .cin(carry[0]), .operation(oper), .result(result[0]), .cout(carry[1]) );

//for loop declaration of ALU1-30
parameter NBIT = 30;
generate
genvar i;
for (i=1; i<=NBIT; i=i+1)
begin u:
alu_top alui( .src1(src1[i]), .src2(src2[i]), .set(setzero), .A_invert(a_in), .B_invert(b_in),
                              .cin(carry[i]), .operation(oper), .result(result[i]), .cout(carry[i+1]) );
end
endgenerate

//diff cout
alu_last alu31( .src1(src1[31]), .src2(src2[31]), .set(setzero), .A_invert(a_in), .B_invert(b_in),
                                                                                  ult(result[31]), .cout(cout) );
```

```
...on Lab2/work/alu.v -- Unsuccessful Compile
vlog -work work -stats=none {F:/NCTU2020spring/Computer_Organization2020/Compute
r Organization Lab2/work/alu.v}
Model Technology ModelSim PE Student Edition vlog 10.4a Compiler 2015.03 Apr  7
2015
-- Compiling module alu
** Error: (vlog-13069) F:/NCTU2020spring/Computer_Organization2020/Computer Orga
nization Lab2/work/alu.v(116): near "alu_top": syntax error, unexpected IDENTIFI
ER, expecting assert  or assume  or restrict .
```

- the get substract res then set method is bugged
  - after the first execution, program would pass through display and end at testbench line 146.

```
VSIM 12> run -all
# ********************************************************
# *              PATTERN RESULT TABLE                    *
# ********************************************************
# * PATTERN *              Result              * ZCV *
# ********************************************************
# * No. 8 error!                                         *
# * Correct result: 00000001     Correct ZCV: 000    *
# * Your result: 0000000X        Your ZCV: x00       *
# ********************************************************
# KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK0dodood
# KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK0doodod
# KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKX0doood0
# KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKXXXXKKKKKKKKKKKKKK0dolx0o
```

- fix by using max(?: syntax) to choose from slt set result and result of substraction

```
Transcript
# Loading work.alu
# Loading work.alu_top
ModelSim> run -all
# ********************************************************
# *              PATTERN RESULT TABLE                    *
# ********************************************************
# * PATTERN *              Result              * ZCV *
# ********************************************************
# *      Congratulation! All data are correct!      *
# ********************************************************
# Correct Count:  9
# ** Note: $finish    : F:/NCTU2020spring/Computer_Organization2020/Computer Organization Lab2/work/testbench.v(146)
#    Time: 205 ns  Iteration: 1  Instance: /testbench
# 1
# Break in Module testbench at F:/NCTU2020spring/Computer_Organization2020/Computer Organization Lab2/work/testbench.v line 146

VSIM 15>
```

Lesson learnt

- **Verilog**
  - what is wire, register
  - what is module
  - verilog syntax(case, for loop macro, etc.)
  - It is a **Hardware Description** Language
  - So think like you are **connecting wires**

- **Modelsim**
  - more familiar with the procesure
  - add to project
  - how to compile
  - what is /work, what is library
  - simlate
  - cmds
    - vsim work.testbench
    - run -all
- **Debugging**
  - docs > everything

2020/4/20 TA hours(about 20:00-~21:50)

- these are from TA's words, my conclusion
- begin end as {}
- always + => -> non-blocking
- always + = -> blocking
  - same time => unpredictable
- if(condition) {} => if condition hold
  - if can only be in always
- implicit mux:
  - :?
  - case
  - if
- slt
  - use mux to choose from result and immediate gen like f(carry)
- for loop
  - check doc is better than asking
- bootcamp, teamviewer, iverilog
- work is like dll(unsure)

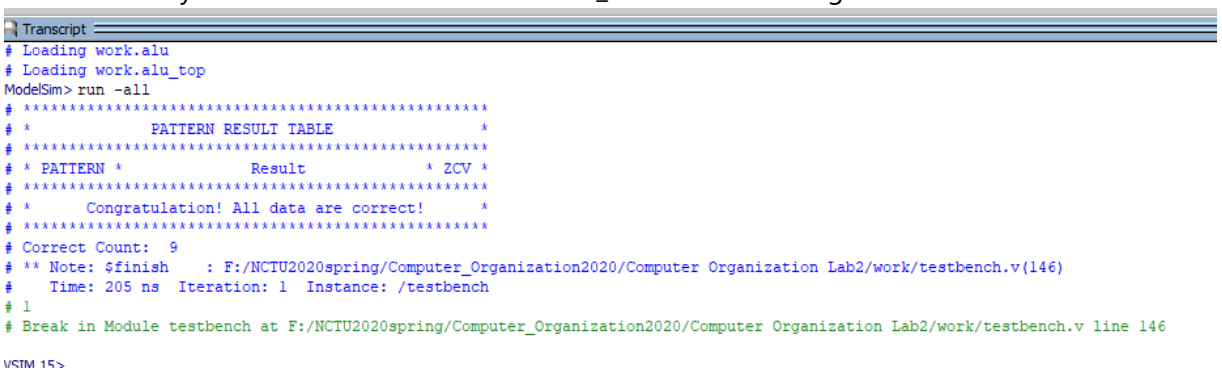
Possible Improvements

- There should be other way to write ALUControl to Ain Bin Opcode
- **Tree structure** of ALU instead of sequential feeding
  - ALU1bit, ALU2bit, ALU4bit, ... ,ALU2^nbit
- **Carry peeking** to add speed
  - would be HARD WORK!!

Some Time Line

- 2020/4/19 started intense coding

- 2020/4/20 TA hours(about 20:00-~21:50)
- 22:18 done! by HDL reference and mask result_t and immediate gen



References

- ALU for loop declaration
  - https://link.springer.com/chapter/10.1007%2F978-1-4615-1713-9_38 (https://link.springer.com/chapter/10.1007%2F978-1-4615-1713-9_38)
  - suggested by TA
  - i think it is similar to C++ macro(i.e. #define)
  - because the error msg related is in the compiling stage and refer to expansion
- Verilog Syntax & Logics
  - https://hackmd.io/@dppa1008/Logic_Design_Mak (https://hackmd.io/@dppa1008/Logic_Design_Mak)

Comment