数字电路实验

作业 1：ALU 的建模及仿真

选课编号：K0166210.02

姓名周子涵选课序号（53）

电子科技大学

通信与信息工程学院

2019 年 11 月

一、verilog 代码（请在代码后附加注解）

```
//------------------------------------------------------------------
//--SAYEH (Simple Architecture Yet Enough Hardware) CPU
//------------------------------------------------------------------
//Arithmetic Logic Unit (ALU)

`timescale 1 ns /1 ns
//宏定义
`define B15to0H 10'b1000000000
`define AandBH    10'b0100000000
`define AorBH     10'b0010000000
`define notBH     10'b0001000000
`define shlBH     10'b0000100000
`define shrBH     10'b0000010000
`define AaddBH    10'b0000001000
`define AsubBH    10'b0000000100
`define AmulBH    10'b0000000010
`define AcmpBH    10'b0000000001

module ArithmeticUnit (
      A, B,
      B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB,
      aluout, cin,    cout, zout
);
//输入输出
input [15:0] A, B;
input B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB;
input cin;
output [15:0] aluout;
output cout,zout;
reg [15:0] aluout;
reg    cout,zout;

      always @(
            A or B or B15to0 or AandB or AorB or notB or shlB or shrB or AaddB or AsubB    or
AmulB    or AcmpB or cin
      )
      begin
            cout = 0; aluout = 0;zout=0;
            case({B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB})
                  //code

                  `B15to0H: begin
```

```
            aluout = B;//输出 B
        end
        `AandBH: begin
            aluout = A&B;//输出 A 与 B
        end
        `AorBH: begin
            aluout = A|B;//输出 A 或 B
        end
        `notBH: begin
            aluout = ~B;//将 B 按位取反
        end
        `shlBH: begin
            aluout = {B[14:0],B[15]};//将 B  左移一位
        end
        `shrBH: begin
            aluout = {B[0],B[15:1]};//将 B 右移一位
        end
        `AaddBH: begin
            {cout,aluout }= A+B+cin;//输出 A+B+cin
        end
        `AsubBH: begin
            {cout,aluout} = A-B-cin;//输出 A-B-cin
        end
        `AmulBH: begin
            aluout = A[7:0]*B[7:0];//输出 A*B
        end
        `AcmpBH: begin
            if(A == B)zout = 1;else zout = 0;
            if (A> B) cout = 1; else cout = 0;//输出 A 比 B
        end
        default: aluout = 0;
    endcase

     end

endmodule
```

二、 testbench 代码

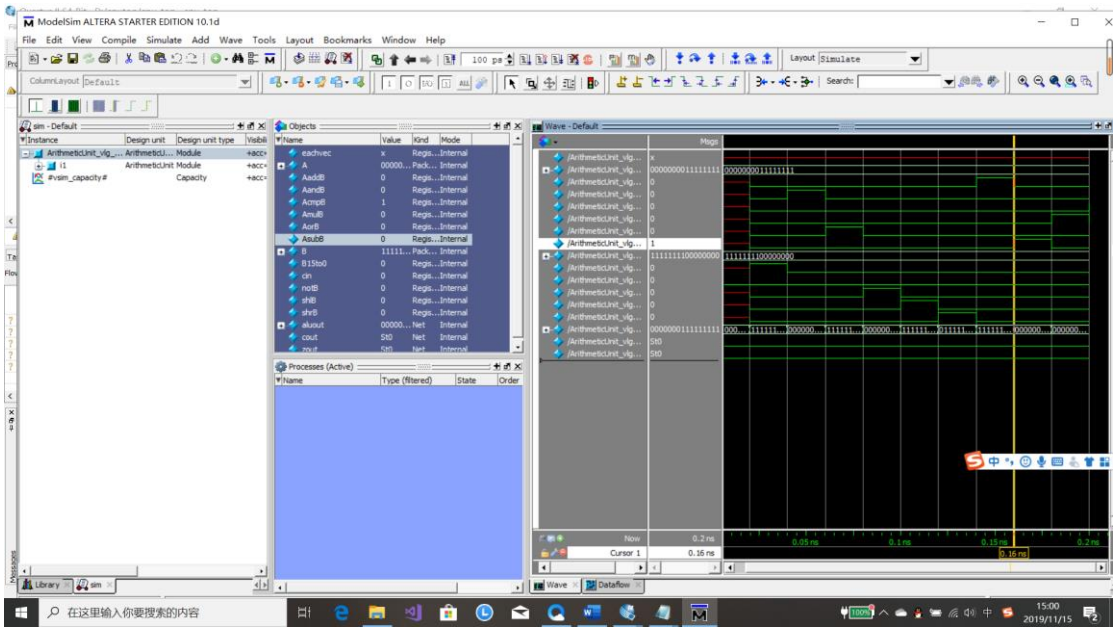
```
// Copyright (C) 1991-2013 Altera Corporation
// Your use of Altera Corporation's design tools, logic functions
// and other software and tools, and its AMPP partner logic
```

```
// functions, and any output files from any of the foregoing
// (including device programming or simulation files), and any
// associated documentation or information are expressly subject
// to the terms and conditions of the Altera Program License
// Subscription Agreement, Altera MegaCore Function License
// Agreement, or other applicable license agreement, including,
// without limitation, that your use is for the sole purpose of
// programming logic devices manufactured by Altera and sold by
// Altera or its authorized distributors.  Please refer to the
// applicable agreement for further details.

// ***************************************************************************
// This file contains a Verilog test bench template that is freely editable to
// suit user's needs .Comments are provided in each section to help the user
// fill out necessary details.
// ***************************************************************************
// Generated on "09/09/2016 16:28:39"

// Verilog Test Bench template for design : ArithmeticUnit
//
// Simulation tool : ModelSim-Altera (Verilog)
//

`timescale 1 ps/ 1 ps
`define B15to0H 10'b1000000000
`define AandBH  10'b0100000000
`define AorBH   10'b0010000000
`define notBH   10'b0001000000
`define shlBH   10'b0000100000
`define shrBH   10'b0000010000
`define AaddBH  10'b0000001000
`define AsubBH  10'b0000000100
`define AmulBH  10'b0000000010
`define AcmpBH  10'b0000000001
module ArithmeticUnit_vlg_tst();
// constants
// general purpose registers
reg eachvec;
// test vector input registers
reg [15:0] A;
reg AaddB;
```

```
reg AandB;
reg AcmpB;
reg AmulB;
reg AorB;
reg AsubB;
reg [15:0] B;
reg B15to0;
reg cin;
reg notB;
reg shlB;
reg shrB;
// wires
wire [15:0]    aluout;
wire cout;
wire zout;

// assign statements (if any)
ArithmeticUnit i1 (
// port map - connection between master ports and signals/registers
      .A(A),
      .AaddB(AaddB),
      .AandB(AandB),
      .AcmpB(AcmpB),
      .AmulB(AmulB),
      .AorB(AorB),
      .AsubB(AsubB),
      .B(B),
      .B15to0(B15to0),
      .aluout(aluout),
      .cin(cin),
      .cout(cout),
      .notB(notB),
      .shlB(shlB),
      .shrB(shrB),
      .zout(zout)
);
initial
begin
A=16'b0000000011111111;
B=16'b1111111100000000;
cin=0;
```

```
        begin
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`B15to0H;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`AandBH;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`AorBH;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`notBH;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`shlBH;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`shrBH;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`AaddBH;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`AsubBH;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`AmulBH;
        #20
        {B15to0, AandB, AorB, notB, shlB, shrB, AaddB, AsubB, AmulB, AcmpB}=`AcmpBH;
        end
end
endmodule
```

三、仿真波形（请对仿真波形进行简要分析）



从上到下依次输出 A,AaddB,AandB,AcmpB,AmulB，AorB,AsubB,B,B15to0,cin,notB,shlB,shrB,aluout,cout,zout 当输入为 A=16'b0000000011111111; B=16'b1111111100000000 时，0.02ns 时，B15to0 上升；0.04ns 时，AandB 上升； 0.06ns 时 AorB 上升；0.08ns 时，AorB 上升；0.1ns 时，shlB 上升；0.12ns 时，shrB，0.14ns 时 AaddB 上升；0.16ns 时，AsubB 上升；0.18ns 时 AmulB 上升；

四、思考题：请简要回答 ALU 的作用是什么？

算术 逻辑运算单元（ALU）的基本功能为加、减、乘、除四则运算，与、或、非、异或 等逻辑操作，以及移位、求补等操作。