Homework 2 –Pipelined ALU

Due Oct. 27, 2015

1. Use Verilog HDL to develop a model of a pipelined ALU with two pipeline stages to support parts of MIPS instruction set. The first pipeline stage performs instruction decoding and data fetch from register file. The second stage performs the corresponding operations. The register file contains 16 registers, each of 32 bits. Your ALU should minimally implement:
   - *Unsigned addition of 32-bit numbers*
   - *2s-complement addition/subtraction of 32-bit numbers*
   - *Logical "and"*
   - *Logical "or"*
   - *Logical "nor"*
   - *Set on less than*

The following figure shows a Verilog code describing some MIPS ALU operations *without pipelining*.

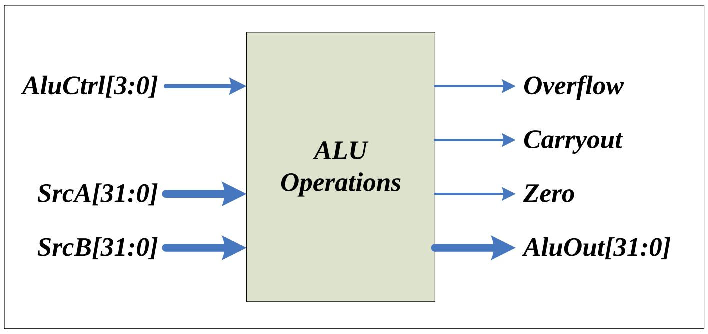
```
module MIPSALU (ALUctl, A, B, ALUOut, Zero);
   input [3:0] ALUctl;
   input [31:0] A,B;
   output reg [31:0] ALUOut;
   output Zero;
   assign Zero = (ALUOut==0); //Zero is true if ALUOut is 0
   always @(ALUctl, A, B) begin //reevaluate if these change
      case (ALUctl)
         0: ALUOut <= A & B;
         1: ALUOut <= A | B;
         2: ALUOut <= A + B;
         6: ALUOut <= A - B;
         7: ALUOut <= A < B ? 1 : 0;
         12: ALUOut <= ~(A | B); // result is nor
         default: ALUOut <= 0;
      endcase
    end
endmodule
```

You need to design a pipelined ALU with two pipeline stages. The first stage implements instruction decoding/register-file-accessing and the second stage implements the corresponding operations. The instruction format shown below contains 4-bit *opcode* field for specifying the operation, 4-bit *srcA* field for source register A, 4-bit *srcB* field for source register B, and 4-bit *dest* field for the destination register where the computation result is to be written back.

opcode	srcA	srcB	dest

opcode = instr[15:12]   srcA=instr[11:8]   srcB=instr[7:4]   dest = instr[3:0]

The following figure shows the second pipelined stage of the ALU. Draw a figure to show the complete design, including the first pipelined stage and the second pipelined stage. Note that some pipelined registers are needed between the two stages. Also, the *AluOut* of the second pipelined stage should be directed to the destination register in the register file.



The following table shows the ALU control lines and the corresponding operation.

ALU Control Line	Function
4'b0000	Logical and
4'b0001	Logical or
4'b0010	Add
4'b0110	Subtract
4'b0111	Set on less than
4'b1100	Logical nor

Write a Verilog code for the pipelined ALU.

2. Verify your RTL codes of the ALU.
3. Use Synopsys Design Compiler to synthesize the RTL ALU design into gate-level netlists based on some standard cell library. What are the critical path delay and total area cost of the ALU? Draw the architecture of the synthesized ALU in your report and explain your observations about the synthesis results.
4. Verify again the synthesized gate-level design using the same test patterns as used in the RTL simulation.
5. Use Xilinx ISE to Synthesize the RTL model of the ALU into gate-level for a selected FPGA chip and verify the design again using the same test patterns used in RTL simulation.

Report Requirements

You can describe the difficulty you encounter and how you solve the problems. Email your report to TA with title: HDL HW2 [your ID] [Your name]. Also upload it to the course website.