

1 Introduction

For this lab, you are going to continue the construction of your simulated computer. The resulting component of this assignment is the control logic for all the other components of the CPU. Additionally, you will create the ALU Control unit that takes control lines and determines which function to tell the ALU to perform.

2 Requirements

The following requirements must be met by your control unit:

1. The assignment will be written in VHDL. The exact signature of the control units should conform to the following definitions:

```
ENTITY Control IS
    PORT(Op: IN STD_LOGIC_VECTOR(31 DOWNTO 26);
         Func: IN STD_LOGIC_VECTOR(5 DOWNTO 0);
         Branch, MemRead, MemWrite, ALUSrc, RegWrite: OUT STD_LOGIC;
         MemToReg, RegDst, Jump, ALUOp: OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
END Control;
```

```
ENTITY ALUControl IS
    PORT(ALUOp: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
         Function: IN STD_LOGIC_VECTOR(5 DOWNTO 0);
         Operation: OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
END ALUControl;
```

2. The control unit will provide 9 outputs:

Register Destination is the line that controls which input is used by the write register of the register file. When it is 0, the input is the Rt value in the instruction; when it is 1, the input is the Rd value in the instruction.

Branch is true if the current instruction could cause a branch to take place.

Jump is true if the current instruction is a jump.

Memory Read is true if the current instruction needs to read a value from the data memory.

Memory Write is true if the current instruction needs to write a value to the data memory.

Memory to Register is the line that controls which input is used by the write data of the register file. When it is 0, the input is the value generated by the ALU; when it is 1, the input is the value generated by the data memory.

ALU Source is the line that controls which input is used by the second operand of the ALU. When it is 0, the input is the value generated by the second data item read from the register file; when it is 1, the input is the sign extended lower 16 bits of the instruction.

Register Write is true if the current instruction stores a value in the register file.

ALU Operation is the set of control lines that feed into the ALU control unit. When they are 00, the ALU should perform addition; when they are 01, the ALU should perform subtraction; and when they are 10, the ALU control unit should use the function input to determine the operation to perform.

3. The control unit will take 2 inputs:

Operation is the 6 most significant bits of the instruction that indicates what functionality the CPU should provide. The set of instructions that must be supported are: `add`, `beq`, `j`, `jal`, `jr`, `lw`, `nor`, `slt`, `sll`, `srl`, `sw`, and `sub`.

Function is the 6 least significant bits of the instruction that indicates what functionality the ALU should provide for an R-type instruction. This is necessary for the control because we need to differentiate between the `jr` and all other R-type instructions. These should directly be passed to the ALU control unit.

4. The ALU control unit will provide 1 output:

Operation is the functionality the ALU should provide during the instruction. Use the same encoding that was specified for the ALU lab.

5. The ALU control unit will take 2 inputs:

ALU Operation is the set of lines from the control unit that indicate how the ALU control unit is to decide which functionality to provide. See the control unit for a description of the various values and their meanings for this input.

Function is the set of 6 least significant bits of the instruction that indicates what functionality the ALU should provide for R-type instructions.

6. All gates that are used must have four or fewer inputs.

7. Each gate should have a 35ps delay.

3 Implementation ideas

You will probably want to develop a simple control structure that looks at the operation and asserts the appropriate control lines. You do not need to generate a complicated finite state machine since you will be doing all the stages in a single clock cycle.

Note that the control unit does not take a clock input. In your final design, the control unit will have to perform the specified operation within a single clock cycle so that it can have the final result ready to the various other units it is controlling so they can complete their operation by the end of the clock cycle. This implies that you cannot be cavalier about how you build your control unit. You must ensure that no extra clock cycles are generated explicitly by your code or implicitly by the simulation of your code.

You need to create a test bench that thoroughly exercises your control unit. You do not need to include a clock for the test bench, but it may be helpful for you to determine how long it takes for the control unit to stabilize on a value. This may be part of the critical path that limits how small you can make the clock period in your final CPU implementation.

When you create your final CPU, you will be connecting the outputs of your control unit to the inputs of all the other parts of your CPU design. You may wish to test that you are able to perform this connection after completing all your other testing. This will not be part of your grade for this assignment, but will make your future assignments easier.

For this assignment, you may want to implement a behavior model for your control unit. This will help simulate the finite state machine that most concisely implements your control unit. It will also ensure you don't need to do your own VHDL version of a PLA.

4 Deliverables

You should turn in an electronic copy of your VHDL, including all the components you created to construct the control units hierarchically. Additionally, you must demonstrate your lab to me at a scheduled time.

The project is worth 100 points as follows:

- 25 points overall structure
 - 10 points good hierarchical design
 - 15 points correct connections between components
- 60 points control unit performs correctly
 - (6 points for each control line)
- 15 points ALU control unit performs correctly
 - (5 points for each input value)