COMPARC Milestone 1

1 Introduction

The machine problem grouping will be composed of two to three members. (Group name format: <pokemon>) Register to a group in your canvas. For the first milestone, we will be implementing two critical blocks for the MIPS processor: the register file, and the Arithmetic and Logic Unit (ALU).,

2 Specifications

You will use the Icarus Verilog and GTKWave in implementing the two blocks. An automated checker will be used for evaluating your work. This requires that you strictly adhere to port naming conventions for each module, which will be discussed in the following subsections. Notes on how to run Icarus Verilog.

iverilog -o <name_of_module>.vvp <name_of_module>.v
vvp <name_of_module>.vvp
gtkwave <name_of_module>.vcd &

2.1 Register File

The MIPS register file contains thirty-two 32-bit registers. It should have two read ports (rd_dataA and rd_dataB), which is addressed using two read address inputs (rd_addrA and rd_dataB, respectively). Reads are asynchronous in nature: when an address is placed in the read address input, the corresponding read port must immediately display (with corresponding logic delays, of course) the contents of the said register, without waiting for the rising edge of the clock.

Writes to the register file are facilitated by one write port (wr_data), which is addressed using a write address input (wr_addr). Unlike reads, writes are registered: data and address ports are set, along with an assertion of an enable signal (wr_en = 1), but the register is not written until the rising edge of the clock signal (elk). When the enable signal is deasserated (wr_en = 0), no writes are performed. All registers, except for R0 (address = 5'dO), may be written. A global reset signal (nrst) is also available and is used to initialize the contents of the register file. When the global reset is asserted (nrst = 0), all registers in the register file are reset to 32'dO. Table 1 shows a summary of all register file ports with their corresponding directions and widths.

Port name	Direction	Width
elk	input	single
nrst	input	single
rd_addrA	input	[4:0]
rd_addrB	input	[4:0]
wr_addr	input	[4:0]
wr en	input	single
wr_data	input	[31:0]
rd_dataA	output	[31:0]
rd_dataB	output	[31:0]

Table 1: Register file port definitions

2.2 ALU

For this exercise, we will only be implementing five operations: addition, subtraction, bitwise AND, bitwise OR, and bitwise NOT. All operations will be performed on 32-bit operands (opA and opB). The operation to be performed will be specified by a select signal (sel = 3'b000 for addition, sel = 3'b001 for subtraction, sel = 3'b010 for AND, sel = 3'b011 for OR, sel = 3'bl00 for NOT). For the subtraction, the operation performed is op A - opB. The ALU is a purely combinational module, with a single 32-bit result output (res). For addition and subtraction, both operands and the result are represented in 2's-complement notation. For the NOT operation, only the first operand is used.

Aside from the result, the ALU will also output flags which describe the result of the current operation. The zero flag will be asserted (z = 1) if the result of the current operation is equal to zero, and deasserted (z = 0) otherwise. The carry flag will be asserted (c = 1) if the result of an addition or subtraction operation has a carry out, and deasserted (c =

0) otherwise. The overflow flag will be asserted (v = 1) if the result of an addition or subtraction operation resulted in arithmetic overflow, and deasserted (v = 0) otherwise. Take note that for bitwise operations, both carry and overflow flags will be deasserted.

In cases where the select signal is set to a value without a corresponding operation, the result and the flags should all be set to zero. Take note that this is the only case when both result and zero flag are both set to zero.

Port name	Direction	Width
op A	input	[31:0]
opB	input	[31:0]
sel	input	[2:0]
res	output	[31:0]
z	output	single
c	output	single
v	output	single

Table 2: ALU port definitions

3 Submission

Grading for this exercise will be broken down as follows:

- Behavioral Model
  - 20 pts register file
  - 5 pts ALU addition
  - 5 pts ALU subtraction
  - 5 pts ALU bitwise operations
  - 5 pts ALU flags

Deadline for submission will be on 23:59 January 28. Send all source Verilog files in a single ZIP file, with your MP group name as the filename of the zip file. Send the file in canvas. Make sure that your codes are working before submission. Since the exercise will be checked automatically, you will not be given a chance to explain whatever is wrong with your code.