Second Partial Project
Delivery 2/3
MIPS I-Type datapath

1 Objectives

To implement the datapath of a custom Microprocessor without Interlocked Pipeline Stage (MIPS) I-Type instruction in SystemVerilog.

2 Second partial grade weight

This delivery constitutes 60% of your second partial final grade.

3 Deadline

23:59 hours on Wednesday October 21st 2020

4 Pre-requisites

It is assumed that you are familiar with working with ModelSim and Quartus. If you require assistance, you can refer to the first assignment tutorial. It is assumed that you have completed the previous assignment for modelling the MIPS R-Type instructions in SystemVerilog.

5 Background

Figure 1 shows the encoding used in I-Type instructions.

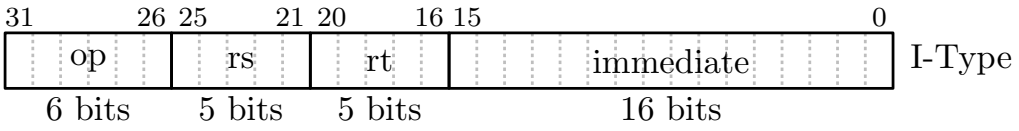

Figure 1: MIPS I-Type instruction encoding.

As shown in, Figure 1, an immediate value is specified directly in the lower half of the I-Type instruction encoding. This immediate value is then used: 1) as an operand in arithmetic or logical operations; 2) as an offset address for load/store instructions using displacement addressing modes; or 3) as the branch address in branch instructions. Field `op` of Figure 1 specifies the instruction to perform, whilst `rs` and `rt` are used as source and destination registers, respectively.

6 Design specifications

The following sections provide an overview of the design specifications for this assignment.

6.1 General specifications

Your custom MIPS design must comply with the following design specifications.

- Instruction set based (but modified) on a standard 16-bit MIPS Microprocessor (μP).
- Single-cycle design.
- Data width is 16-bits.
- Instruction encoding (instruction width) is 32-bits long.
- Register File (RF) contains 32 registers.
  - `r0` **must** always be 0, *i.e.*, `r0` is not writeable.
  - `r31` is return address.
- Instruction Memory (IM) is parameterizable.
- Data Memory (DM) is parameterizable.

6.2 I-Type top-level

Table 1 specifies the top-level ports required for this assignment.

Table 1: Top-level ports.

Port name	Direction	Width	Description
`clk`	`input`	1	Clock signal
`asyn_n_rst`	`input`	1	Asynchronous active-low reset
`instruction`	`input`	32	Encoded instruction to be performed
`pc`	`input`	8	Current PC value
`next_pc`	`output`	8	PC value to be loaded in next clock cycle

This top-level module consists only input `clk`, `asyn_n_rst`, `instruction` and `pc` which correspond to the clock signal, an asynchronous active-low reset, R- and I-type instructions to be performed, and the current value of Program Counter (PC), respectively. The only output of this module is `next_pc`, which represents the value of PC to be loaded during the next clock cycle.

6.3 List of I-Type instructions

Your design must be able to perform the I-Type instructions specified in Table 2 as well as **all** R-Type instructions.

Table 2: I-Type operations.

op value	Instruction	Syntax	Meaning
6'b001000	ADDI	ADDI rt, rs, imm	Reg[rt] ← Reg[rs] + imm
6'b110000	SUBI	SUBI rt, rs, imm	Reg[rt] ← Reg[rs] - imm
6'b110001	NANDI	ANDI rt, rs, imm	Reg[rt] ← ∼(Reg[rs] & imm)
6'b110010	NORI	ORI rt, rs, imm	Reg[rt] ← ∼(Reg[rs] | imm)
6'b110011	XNORI	XORI rt, rs, imm	Reg[rt] ← ∼(Reg[rs] ^ imm)
6'b001100	ANDI	ANDI rt, rs, imm	Reg[rt] ← Reg[rs] & imm
6'b001101	ORI	ORI rt, rs, imm	Reg[rt] ← Reg[rs] | imm
6'b001110	XORI	XORI rt, rs, imm	Reg[rt] ← Reg[rs] ^ imm
6'b001111	LUI	LUI rt, imm	Reg[rt] ← {imm[7:0], 8'b0}
6'b110100	LLI	LLI rt, imm	Reg[rt] ← {8'b0, imm[7:0]}
6'b110101	LI	LI rt, imm	Reg[rt] ← imm
6'b100011	LW	LW rt, imm(rs)	Reg[rt] ← Mem[rs+imm]
6'b101011	SW	SWR rt, imm(rs)	Mem[rs+imm] ← Reg[rt]
6'b000100	BEQ	BEQ rt, rs, imm	if (Reg[rs] == Reg[rt]) then PC ← imm
6'b000101	BNE	BNEQ rt, rs, imm	if (Reg[rs] != Reg[rt]) then PC ← imm
6'b000110	BLEZ	BLEZ rs, imm	if (Reg[rs] <= 0) then PC ← imm
6'b000111	BGTZ	BGTZ rs, imm	if (Reg[rs] > 0) then PC ← imm

You are required to implement a DM in order to perform `LW` and `SW` instructions. For this deliverable, DM should contain data 64 addresses.

6.4 SystemVerilog design files

The minimum required SystemVerilog design files are listed below. You may decide to create new SystemVerilog modules and files in order to create different hierarchy levels. For example, you may decide to create a SystemVerilog module specifically for instruction decoding inside the control unit.

- `mips_pkg.sv`. Package file common to all design files. You must declare all the necessary parameters and data types in this file.
- `itype.sv`. I-Type top-level module.
- `alu.sv`. Arithmetic and Logic Unit (ALU) module description.
- `sgn_ext.sv`. Zero/sign extender module description.
- `mux_2x1.sv`. 2x1 Multiplexer (MUX) module description.
- `control_unit.sv`. Control unit module description.
- `dm.sv`. DM module description.

7 Grading criteria

The following grading criteria will be considered.

1. **The correct functionality of your designs.** I will use my own testbenches in order to automatically stress your designs and verify that they perform the tasks according to the specifications. For example, I will try different values for the parameters in your designs and I expect them to still perform according to the specifications. This is why it is paramount that you follow the name convention specified for file names and port names. Moreover, it is important that your designs and testbenches compile in ModelSim without errors. **Your maximum grade for this assignment will automatically drop to 50/100 should ModelSim trigger a compilation or simulation error.**
2. **The quality of your testbenches.** Even though I will use my own testbenches, I expect you to consider a thorough and concious set of test scenarios. In this way, you should be able to spot any mismatches between the expected results and the actual results provided by your designs.
3. **Your designs must be synthesized in Quartus without latches and without errors.** Warnings are tolerated at this point. **Your maximum grade for this assignment will automatically drop to 50/100 should Quartus trigger a synthesis error or generate unwanted latches.** Remember that a design is not useful if it can't be synthesized.

8 Deliverables and Submission instructions

Prepare a single `zip` file containing **at least** the following files. You may have created additional SystemVerilog modules as stated in Section 6.4. If that's the case, please include them in your `zip` file.

1. `mips_pkg.sv`.

2. `itype.sv`.
3. `tb_itype.sv`. Testbench for your top-level I-Type design.
4. `alu.sv`.
5. `sgn_ext.sv`.
6. `mux_2x1.sv`.
7. `control_unit.sv`.
8. `itype.do` and all ModelSim scripts for compiling and simulating your design.
9. A Quartus Register-Transfer Level (RTL) screenshot of your synthesized top-level I-Type design.

Submit your assignment through Canvas **no later** than 23:59 hours on Wednesday October 21st 2020.
Please send any questions to isaac.perez.andrade@tec.mx.