Second Partial Project
Delivery 3/3
Complete MIPS datapath

1 Objectives

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

2 Second partial grade weight

This delivery constitutes 5% of your second partial final grade.

3 Deadline

23:59 hours on Wednesday October 28th 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 and I-Type instructions in SystemVerilog.

5 Specifications

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

5.1 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) contains 256 address.
- Data Memory (DM) contains 256 address.

5.2 MIPS 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

This top-level module consists of only inputs `clk` and `asyn_n_rst` and should be named `mips`. There are no outputs in this top-level module. You must include the IM and the Program Counter (PC) as part of your MIPS design.

5.3 List of instructions

Table 2 shows the minimum set of instructions that your MIPS should support. You may support additional instructions that you deem necessary.

Note that instructions highlighted in `blue` correspond to the new instructions to be added to your existing `itype` design.

Table 2: Required MIPS instructions.

Instruction	Syntax	Meaning
	R-Type	
ADD	ADD rd, rs, rt	Reg[rd] ← Reg[rs] + Reg[rt]
SUB	SUB rd, rs, rt	Reg[rd] ← Reg[rs] - Reg[rt]
NAND	NAND rd, rs, rt	Reg[rd] ← ∼(Reg[rs] & Reg[rt])
NOR	NOR rd, rs, rt	Reg[rd] ← ∼(Reg[rs] | Reg[rt])
XNOR	XNOR rd, rs, rt	Reg[rd] ← ∼(Reg[rs] ^ Reg[rt])
AND	AND rd, rs, rt	Reg[rd] ← Reg[rs] & Reg[rt]
OR	OR rd, rs, rt	Reg[rd] ← Reg[rs] | Reg[rt]
XOR	XOR rd, rs, rt	Reg[rd] ← Reg[rs] ^ Reg[rt]
SLL	SLL rd, rs, sa	Reg[rd] ← Reg[rt] << sa
SRL	SLL rd, rs, sa	Reg[rd] ← Reg[rt] >> sa
SLA	SLL rd, rs, sa	Reg[rd] ← Reg[rt] <<< sa
SRA	SLL rd, rs, sa	Reg[rd] ← Reg[rt] >>> sa
JR	JR rs	PC ← Reg[rs]
	I-Type	
ADDI	ADDI rt, rs, imm	Reg[rt] ← Reg[rs] + imm
SUBI	SUBI rt, rs, imm	Reg[rt] ← Reg[rs] - imm
NANDI	ANDI rt, rs, imm	Reg[rt] ← ∼(Reg[rs] & imm)
NORI	ORI rt, rs, imm	Reg[rt] ← ∼(Reg[rs] | imm)
XNORI	XORI rt, rs, imm	Reg[rt] ← ∼(Reg[rs] ^ imm)
ANDI	ANDI rt, rs, imm	Reg[rt] ← Reg[rs] & imm
ORI	ORI rt, rs, imm	Reg[rt] ← Reg[rs] | imm
XORI	XORI rt, rs, imm	Reg[rt] ← Reg[rs] ^ imm
LUI	LUI rt, imm	Reg[rt] ← {imm[7:0], 8'b0}
LLI	LLI rt, imm	Reg[rt] ← {8'b0, imm[7:0]}
LI	LI rt, imm	Reg[rt] ← imm
LW	LW rt, imm(rs)	Reg[rt] ← Mem[Reg[rs]+imm]
SW	SWR rt, imm(rs)	Mem[Reg[rs]+imm] ← Reg[rt]
BEQ	BEQ rt, rs, imm	if (Reg[rs] == Reg[rt]) then PC ← imm
BNE	BNEQ rt, rs, imm	if (Reg[rs] != Reg[rt]) then PC ← imm
BLEZ	BLEZ rs, imm	if (Reg[rs] <= 0) then PC ← imm
BGTZ	BGTZ rs, imm	if (Reg[rs] > 0) then PC ← imm
	J-Type	
J	J imm	PC ← imm
JAL	JAL imm	Reg[31] ← PC+1 PC ← imm

5.4 SystemVerilog design and testbench files

The minimum required SystemVerilog design files and testbenches 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.

- Desing units.
  - `mips_pkg.sv`. Package file common to all design files. You must declare all the necessary parameters and data types in this file.
  - `mips.sv`. Top-level MIPS module.
  - `alu.sv`. Arithmetic and Logic Unit (ALU) module description.
  - `rf.sv`. RF module description.
  - `sgn_ext.sv`. Zero/sign extender module description.
  - `mux_2x1.sv`. 2x1 Multiplexer (MUX) module description.
  - `mux_4x1.sv`. 4x1 MUX module description (if needed).
  - `mux_8x1.sv`. 8x1 MUX module description (if needed).
  - `control_unit.sv`. Control unit module description.
  - `dm.sv`. DM module description.
  - `im.sv`. IM module description.
  - `program_counter.sv`. PC module description.
- Testbenches.
  - `tb_mips.sv`. Top-level MIPS testbench. This testbench tests all possible MIPS instructions.
- IM initialization file. This files may be in either binary or hexadecimal format.
  - `mips_tests.txt`. File for initializing IM in `tb_mips.sv`.

6 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 warnings and without errors.
   (a) **Each RTL warning will deduct 5% of your final project grade.**
   (b) **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, without RTL warnings and without errors.**

(a) **Each RTL warning will deduct 5% of your final project grade.**

(b) **Your maximum grade for this assignment will automatically drop to 50/100 should Quartus trigger a synthesis error or generate unwanted latches.**

The only warnings that will be tolerated are those that are not related to the RTL itself. Examples of such warnings are:

- `Warning (18236): Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PARALLEL_PROCESSORS in your QSF to an appropriate value for best performance.`
- `Warning (13046): Tri-state node(s) do not directly drive top-level pin(s).`
- `Warning (292013): Feature LogicLock is only available with a valid subscription license. You can purchase a software subscription to gain full access to this feature.`
- `Warning (15714): Some pins have incomplete I/O assignments. Refer to the I/O Assignment Warnings report for details.`

Remember that a design is not useful if it can't be synthesized.

7 Deliverables and Submission instructions

Prepare a single `zip` file containing the following items.

1. **All your design, testbench and memory initialization files.** You may follow the list provided in Section 5.4 as a reference. You may have created additional SystemVerilog modules as stated in Section 5.4. If that's the case, please include them in your `zip` file.

2. `mips.do`. A ModelSim script for compiling and simulating your design. This will be used for testing the J-Type instructions, along with the R-Type and I-Type instructions. You may use several `.do` files for different tasks, for example, one `.do` for creating a library, one for compiling, one for simulating, one for adding waveforms and one for running all tasks at once.

3. A computer-drawn schematic diagram showing your final Microarchitecture (μA) of your MIPS design.

   (a) This μA must support all R-Type, I-Type and J-Type MIPS instructions.

   (b) All signals must be clearly labelled, including bus widths.

4. Design synthesis. A screenshot of the Quartus-generated Register-Transfer Level (RTL) view.

Submit your assignment through Canvas **no later** than 23:59 hours on Wednesday October 28th 2020. Only one submission per team is necessary.
Please send any questions to isaac.perez.andrade@tec.mx.