Experiment 1

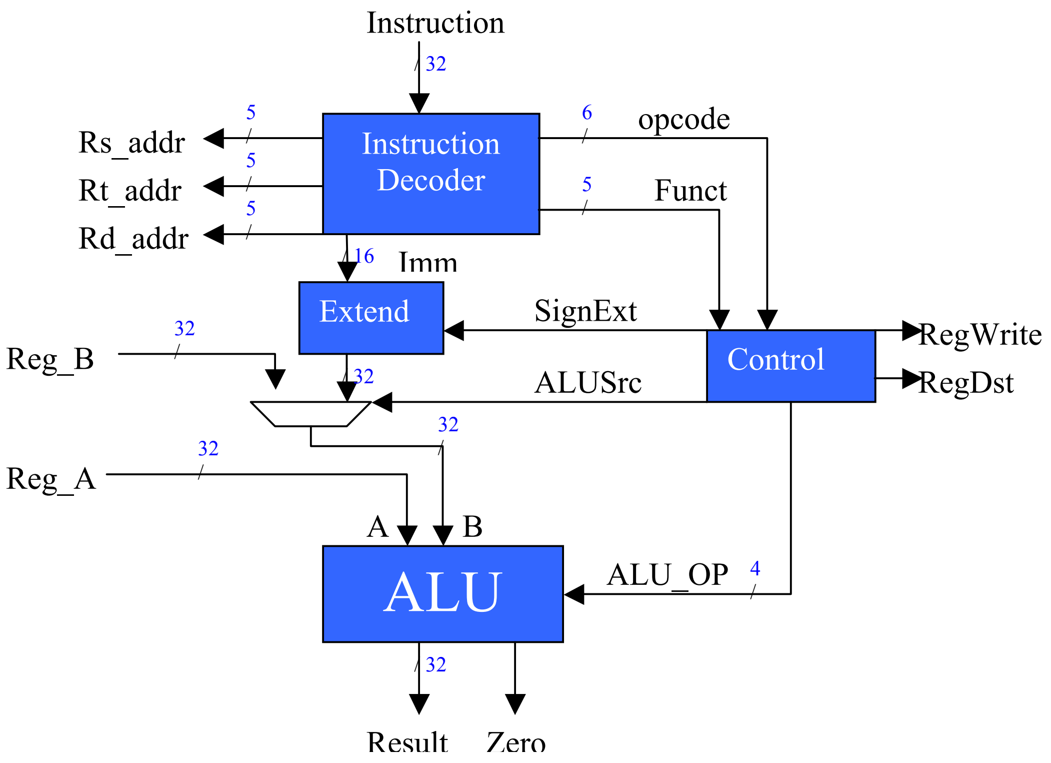
Part 1 (4 points)

Design and implement ALU (Arithmetic Logic Unit), Instruction Decoder, and Basic Control unit of a single-cycle MIPS processor, capable of performing the following types of instructions:

R-type: ADD, AND, OR, SUB, XOR, SLT, SLTU

I-type: ADDI, ANDI, ORI

The ALU, Instruction Decoder, and Control should be connected as shown in the diagram below:



The names, widths, and meanings of all inputs and outputs of both units are given in the tables below:

<u>ALU:</u>

Name	Mode	Width	Meaning
A	In	32	The Value of Reg_A
B	In	32	The value of Reg_B or immediate
ALU_OP	In	4	Specific instruction as to the ALU operation. See the table of Operation codes below.
Result	Out	32	Result of a current operation
Zero	Out	1	Value of the Zero flag after the execution of a current instruction

Instruction Decoder:

Name	Mode	Width	Meaning
Instruction	In	32	Instruction word
Rs_addr	Out	5	Address of register Rs
Rt_addr	Out	5	Address of register Rt
Rd_addr	Out	5	Address of register Rd
Opcode	Out	6	The opcode for the instruction. See the table of Operation codes below.
Funct	Out	6	Specific

			instruction to be executed for the given opcode. See the table of Operation codes below.
Imm	Out	16	immediate field

Control:

Name	Mode	Width	Meaning
Opcode	In	6	The opcode for the instruction. See the table of Operation codes below.
Funct	In	6	Specific instruction to be executed for the given opcode. See the table of Operation codes below.
RegDst	Out	1	Which register address to write to ‘1’ for Rd ‘0’ for Rt
RegWrite	Out	1	‘1’ if instruction calls for the result to be stored in register. Else ‘0’
ALU_Src	Out	1	Which input should go to the ALU ‘0’ for Reg_A ‘1’ for extImm
SignExt	Out	1	‘1’ if the Imm value should be Sign-Extended
ALU_OP	Out	4	Specific

			instruction as to the ALU operation. See the table of Operation codes below.

Table of Operation codes:

Type	Opcode (hex)	Funct (hex)	Instruction	ALU_OP (bin)
R	0	20	ADD	0010
	0	22	SUB	0110
	0	24	AND	0000
	0	25	OR	0001
	0	26	XOR	1101
	0	2A	SLT	0111
	0	2B	SLTU	1110
I	8	X	ADDi	0010
	C	X	ANDi	0000
	D	X	ORi	0001

The ALU, Instruction Decoder, and Control must be described using two different design styles:

- **ALU and Instruction Decoder - dataflow description** (concurrent statements only)
- **Control - synthesizable behavioral description** (based on a process and case statements).

As a part of the design process for the dataflow description:

1. Draw an optimized block diagram of the ALU using medium scale logic components, such as multiplexers, adders, shifters, rotators, 32-bit bitwise AND, etc. Please clearly mark the widths and directions of all buses.

2. Translate your block diagram into a synthesizable VHDL dataflow model (using only concurrent statements).

In the lab report include:

1. Block diagram of the ALU (hand-drawn hardcopy submitted in class and used during an exit quiz, and an electronic copy submitted using Blackboard in the pdf or MS Word format (preferably, the scanned version of the hand-drawn hardcopy).
2. VHDL source codes for the circuit description (electronic versions submitted using Blackboard)
3. Testbench and waveforms from the functional simulation (electronic versions submitted using Blackboard).

Part 2 (2 bonus points)

Extend the instruction set of the functional unit implemented in Part 1 with the following operations Shift Left Logical (SLL), Shift Right Logical (SRL), Shift Left Logical Variable (SLLV), and Shift Right Logical Variable (SRLV):

Type	Opcode (hex)	Funct (hex)	Instruction	ALU_OP (bin)
	0	0	SLL	1110
	0	1	SRL	1111
	0	4	SLLV	1000
	0	6	SRLV	1001

For SRLV and SLLV the shift amount is specified in the register at address Rs and for SRL and SLL the shift amount is stored in the shamt field of the instruction.

Test the circuit implementing both operations first, using a separate testbench.

Then, integrate your design with the design for Part 1.

As a part of the design process:

1. Draw an optimized block diagram of the variable rotator/shifter using medium scale logic components, such as multiplexers, one-bit shifters, etc .
2. Translate your block diagram into two alternative VHDL descriptions
   a. dataflow only model (using only concurrent statements)
   b. structural model at the same level of abstraction as your block diagram. Use dataflow model to describe each individual medium-scale component appearing in your block diagram.
3. Write a separate testbench capable of verifying the functionality of both of your descriptions for all instructions and randomly chosen inputs.

In the lab report include:

1. Optimized block diagram of the variable shifter/rotator (hand-drawn hardcopy submitted in class and used during an exit quiz, and an electronic copy submitted using Blackboard in the pdf or MS Word format (preferably, the scanned version of the hand-drawn hardcopy).
2. VHDL source codes for:
   a. variable rotator and shifter alone, using both types of the circuit description (dataflow and structural)
   b. integrated design from Parts 1 and 2
   (electronic versions submitted using Blackboard)
3. Testbench and waveforms from the functional simulation (electronic versions submitted using Blackboard) for
   a. variable rotator and shifter alone
   b. integrated design from Parts 1 and 2.

Important Dates

	Monday section	Tuesday section	Wednesday section	Thursday section
Hands-on Session and Introduction to the Experiment	01/25/2010	01/26/2010	01/27/2010	01/28/2010
Demonstration and Deliverables Due	02/01/2010	02/02/2010	02/03/2010	02/03/2010

Please email your suggestions to Kris Gaj.