CS2204 ProjectE

8 bit Processor (Register-Register Architecture)

This project will involve designing a 8 bit fixed point ALU with support for a fixed set of instructions(ADD,SUB,SLL,XOR etc).

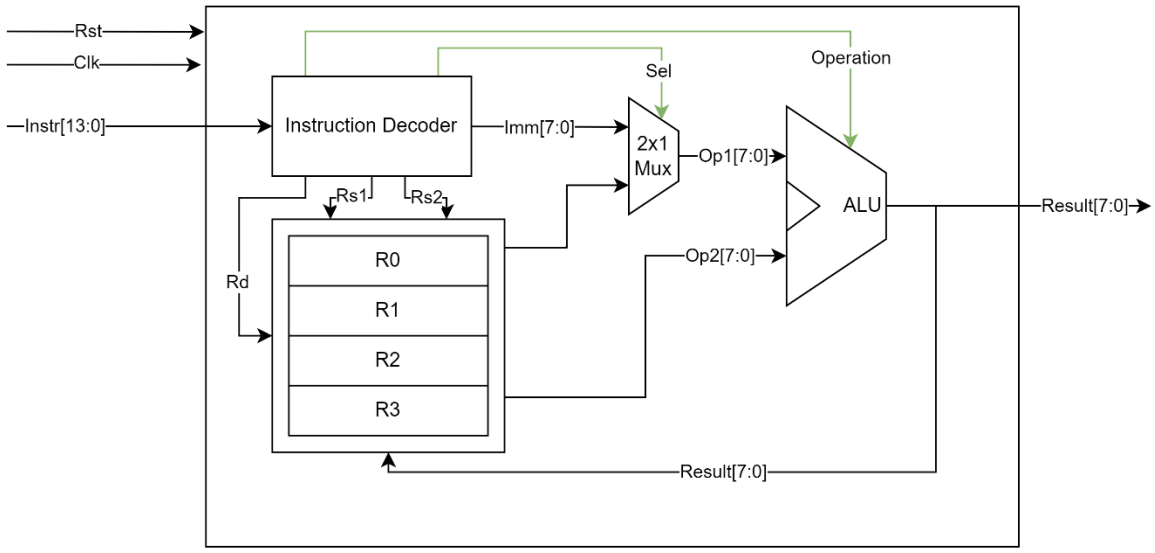

fig 1

ALU: An arithmetic logic unit is a combinational digital circuit that performs arithmetic and bitwise operations on binary numbers. All the microprocessor/microcontrollers have an ALU inside them which is one of the most critical parts.

Theory

Every Microprocessor having a specific architecture understands a fixed set of instructions using which it can perform pretty much any of the complex tasks or programs. This set of instructions is called as the ISA or Instruction Set Architecture of the processor.

There are a number of different architectures such as the x86 architecture by intel or the ARM architecture etc. For the purpose of this project we will design a custom architecture processor with a fixed set of instructions defined in the following sections.

We have worked with FPGAs for CS2204 which have a reconfigurable architecture and can emulate or function as pretty much any digital circuit. A microprocessor on the other hand has a fixed architecture but can still perform most of the functions. The task to be performed at any instant of time is upto the programmer who writes the instructions that tells the processor to perform certain functions on the data. These instructions are written in **assembly language**.

Hint: Your top module should look something as follows.

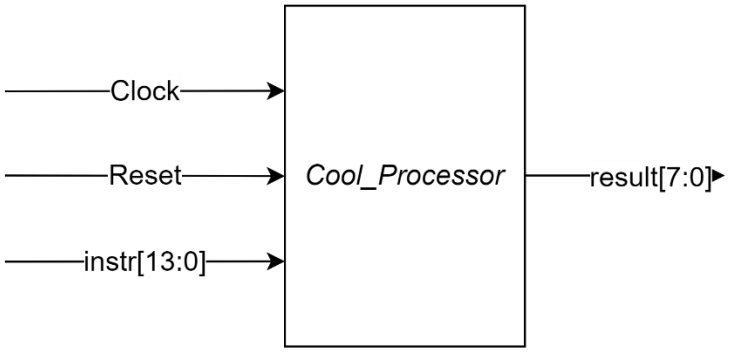

Fig 2

The block diagram *fig 1* illustrates the basic components involved in the design. **All of these components comprise of Multiplexers, Decoders etc which we have come across in the class as well in the lab exercises.**

We will now briefly review each of the 4 components shown above

Instruction Decoder

This component gets all the information about the 14 bit assembly instruction and translates this instruction to meaningful signals that all other components use.

(Question: From all the basic digital component we learnt in class, which component do you think this module will basically be???)

Register File

This is a simply a set of 4 registers which hold the data the processor operates on. Each register is 8 bits wide.

(Question: How many bits would be required to uniquely identify each of the 4 registers???)

ALU

An arithmetic logic unit is a combinational digital circuit that performs arithmetic and bitwise operations on two binary numbers. The function it is supposed to perform depends on the **Operation Code** or **Opcode** defined in the 14 bit instruction.

(Question: If the instruction has 4 bits for the opcode, how many unique operations can this ALU perform???)

2x1 Mux

This component is used to select the data source for the ALU to operate on.

(Question: How many select lines would be required for this component? Where would the select line information come from???)

Instruction Set

The custom set of instructions that should be supported by the ALU are as follows

Mnemonic	Opcode	Function
AND	0x0	Rd = Rs1 and Rs2
OR	0x1	Rd = Rs1 or Rs3
NOT	0x2	Rd = not Rs1
XOR	0x3	Rd = Rs1 xor Rs5
INC	0x4	Rd = Rs1 + 1
DEC	0x5	Rd = Rs1 - 1
SLL	0x6	Rd = Rs1 << 1
SRL	0x7	Rd = Rs1 >> 1
ADD	0x8	Rd = Rs1 + Rs2
SUB	0x9	Rd = Rs1 - Rs2
LOAD	0xA	Rd = immediate
OUT	0xB	Result = Rd
NOP	0xC	Do nothing!!!
	0xD	
	0xE	
	0xF	

Table 1

The 14 bit instruction consists of all the information required for the ALU to function including the source, destination as well as the operation to be performed.

The instruction format for all of the above instructions looks something as follows.

Instruction	Bits				
	13:10	9:8	7:6	5:4	3:0
Arith/Logical	Opcode	Rd	Rs1	Rs2	xx
Data Transfer	Opcode	Rd	Immediate		

Eg. If we want the processor to add two 8 bit numbers stored in registers R1,R2 and store the result in R0, the instruction to do that would look something as follows.

ADD R0,R1,R2 = $(118)_H$

Instruction	Bits				
	13:10	9:8	7:6	5:4	3:0
Arith/Logical	0000	00	01	10	xx

Your design must be able to able to support all the instructions shown in *Table 1*. You are encouraged to add your own custom instrcutions for opcodes 0xD, 0xE and 0xF!!!

Verification: All the individual components as well the integrated module will be tested using test vectors by reading inputs from a .txt file or by using a testbench. I will provide a testbench for the entire module which will help you debug and validate your design.

This verification requires us to test the design by taking inputs from files. This is a verification method used heavily in the industry and great way to enhance your testing skills.

This part was not covered in the coursework and will be taken up during a zoom session whenever requested after the design is complete.

Milestones:

- Understand the ALU design and identify the input/outputs for each of the modules
- Source and testbench files for each of the individual modules involved in the design
    1. ALU
    2. Instruction Decoder
    3. 2x1 Mux
    4. Register File
- Integrate the modules
- Write your own program on the ALU you designed to do simple things such as an 8 bit counter or a program to generate Fibonacci series etc!!!