Assignment 7 Part 3 Group 27

Verilog Assignment 7

Problem No.: 3

Semester: 5

Group No.: 27

Yash Sirvi (21CS10083)

Sanskar Mittal (21CS10057)

Question:

Implement the register bank and integrate the same with the ALU module as designed earlier. Write a top-level module for testing the modules by implementing operations like:
Rx = Ry op Rz
where Rx, Ry, Rz and op can be specified from outside.

Inputs:

- 5 bit register addresses: **rs**,**rt** and **rd**
- 4 bit opcode (for ALU operations)
- 32 bit input (for storing values in the register)
- For FPGA implementation, there is only one 16 bit input **in**

Functions:

Opcode

- 0001 -> ADD
- 0010 -> SUB
- 0011 -> AND
- 0100 -> OR
- 0101 -> XOR
- 0110 -> NOT

- 0111-> SLA
- 1000 -> SRA
- 1001 -> SRL

Register Bank:

We have implemented **16** general-purpose 32-bit registers in the register bank.
Moreover one 32-bit program counter and 32-bit stack pointer are also used.

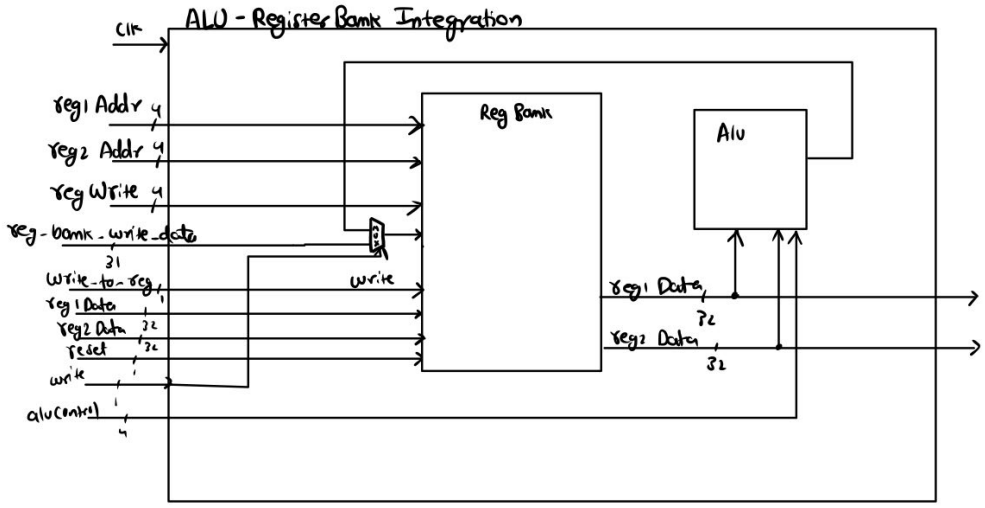
ALU integration:

Based on the opcode provided, the corresponding operations are carried out for the values in the given registers and the output is stored in the destination register. This is implemented in an ALU module which is implicitly used in the main module functionality.

FPGA Implementation:

- We have a separate take_input() module for taking input from FPGA
- Since FPGA has limited input switches, we use a single 16 bit **in** to get input, implemented via a state machine
- We then assign the input to given registers according to the current state
- Sinit_reg1: we first initialise the value of some registers. This state takes its input as the address of the register we want to initialise
- Sinit11: takes the first 16 bit input value
- Sinit12: takes the next 16 bit input value
- Sinit_reg2: address of 2nd register to initialise
- Sinit21: takes first 16 bit input
- Sinit22: takes the next 16 bit input
- Sreg_addr: in[15:11] stores rs address, in[10:6] stores rt address, in[5:1] stores rd address
- Sfunc: in[3:0] stores the function opcode
- Sread1: outputs the value of the final operation's lower 16 bits
- Sread2: outputs the higher 16 bits
- **nextstate** helps us move between states manually
- **lock** allows us to change the **state** at most once every press. Avoids multiple state changes if the **nextstate** button is pressed for too long

Schematic


ALU - Register Bank Integration
clk
reg1 Addr
reg2 Addr
reg Write
reg_bank_write_data
write_to_reg
reg1 Data
reg2 Data
reset
write
alu(cntrl)
Reg Bank
write
ALU
reg1 Data
reg2 Data