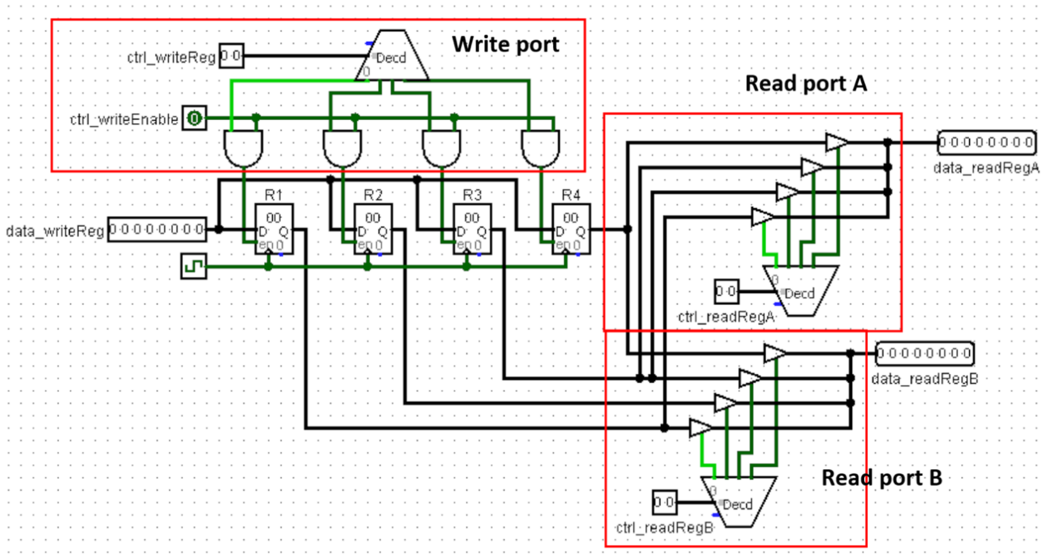
Description of My Design Implementation

- ECE 550 Project Checkpoint3
- Name: Rui Cao
- netID: rc384

1. Overall Design

- Decoder which converts 5 bits binary to 2^5 bits one hot: decoder32.v
- Read port based on decoder: read_port.v
- Write port based on decoder: write_port.v
- 1-bit DFFE: dffe.v
- 32-bit register based on DFFE: register32.v
- Register file: regfile.v
    - includes 32 32-bit registers
    - and also includes 2 read ports and 1 write port
- The circuit diagram below (provided by our Checkpoint 3 PDF) shows my design idea. Although it only contains 4 8-bit registers, its design idea is consistent with the register file including 32 32-bit registers. We just need to increase the number of registers in the diagram to 32 and use 32-bit registers and use 5-32 decoder rather than 2-4 decoder. According to this circuit diagram, we can design the circuit diagram of the register file including 32 32-bit registers smoothly and quickly. Everything is very clear.



2. Detailed Description

2.1. Decoder:

This decoder is designed for converting 5 bits binary to 2^5 bits one hot. The design of the decoder circuit starts with a truth table, and then I will represent the behavior using Boolean expressions (SoP), finally translate Boolean expression into circuit. This process is shown in the following two figures (from our PPT). Although figures below show the design process of the 2-4 decoder circuit, we need a 5-32 decoder. The principles of design process remain the same.

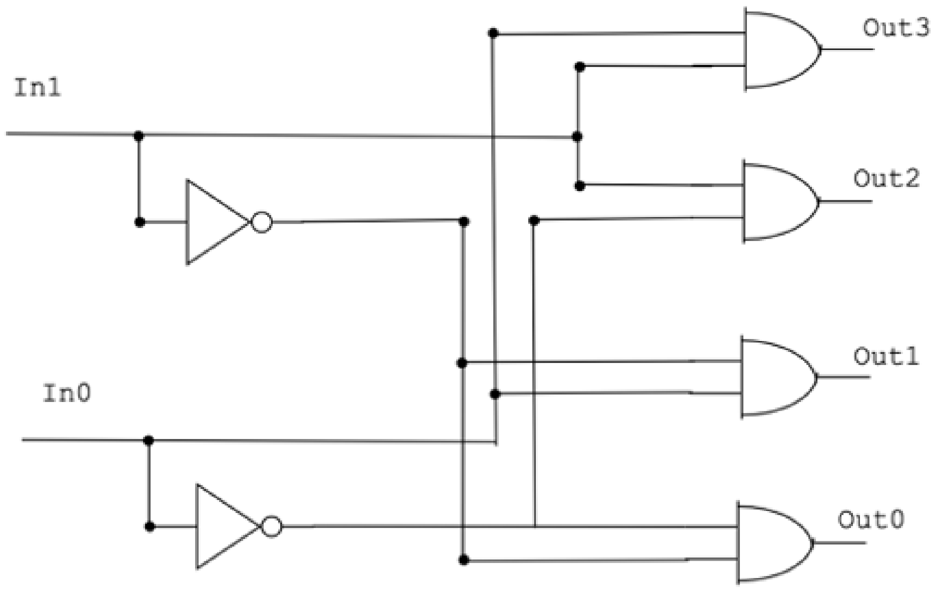
- Start with a truth table
  - Now input unconstrained

In1	In0	Out3	Out2	Out1	Out0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

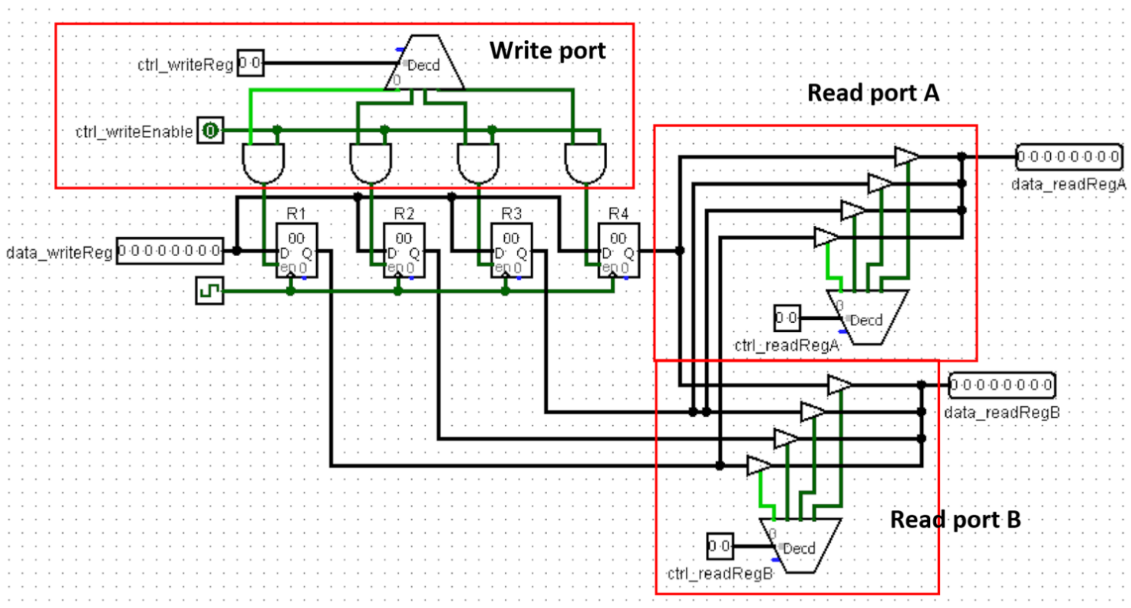
- SoP for each of the 4 outputs:
  - Out3 = In1 and In0
  - Out2 = In1 and (Not In0)
  - Out1 = (Not In1) and In0
  - Out0 = (Not In1) and (Not In0)

2:4 Decoder

- 2-to-4 decoder

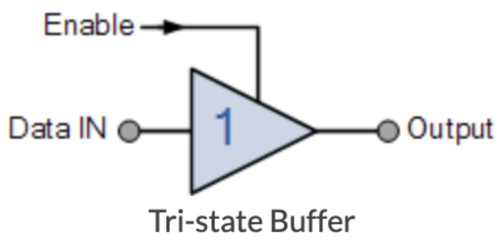


2.2. Read Port:

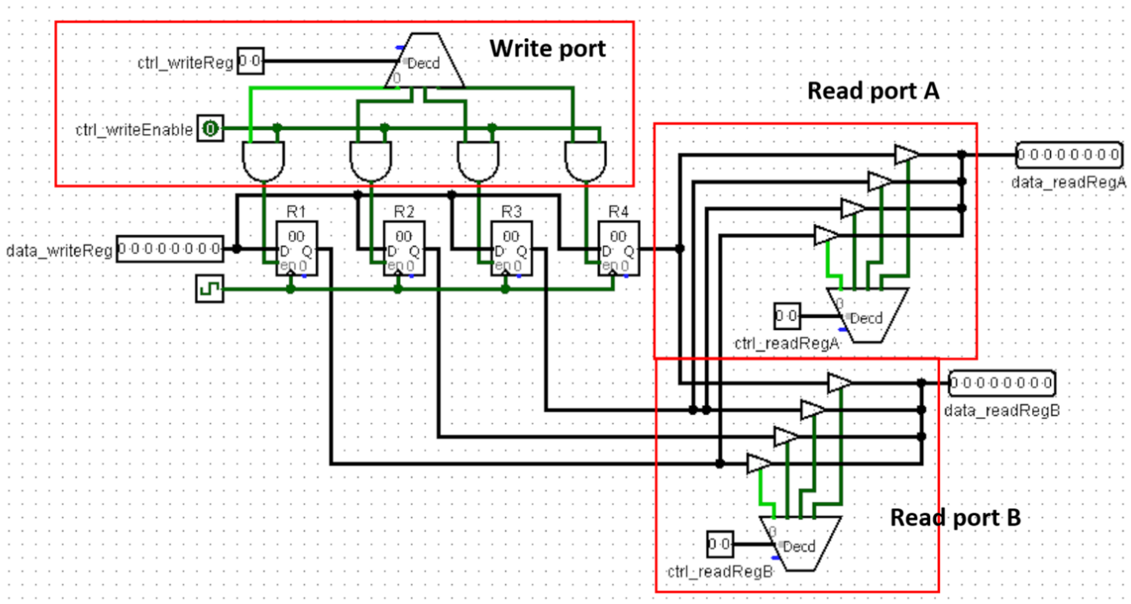


- The parts marked "Read port A" and "Read port B" in the above diagram are the 2 read ports contained in the register file. In this project, we need 32 registers, so there will be 32 outputs of decoder and 32 outputs of 32 registers (q0, q1, q2, q3...q31). This means that the 4 black wires and 4 green wires in the read port part of the circuit diagram above should be 32 wires in this project. The output of my read_port module is the data stored in one of the 32 32-bit registers, which is what we want to read from the register file. The inputs of my read_port module are ctrl_readReg (ctrl_readRegA or ctrl_readRegB) which determines which register we want to read and 32 outputs of 32 registers.
- Tristate buffers: The following figure shows the truth table of tristate buffers. These 32 tristate buffers can help us choose what we want to read from the 32 outputs (q0, q1, q2, q3...q31) of 32 registers based on ctrl_readReg

(ctrl_readRegA or ctrl_readRegB).

Symbol	Truth Table		
	Enable	IN	OUT
	0	0	Hi-Z
	0	1	Hi-Z
	1	0	0
	1	1	1
Read as Output = Input if Enable is equal to “1”			

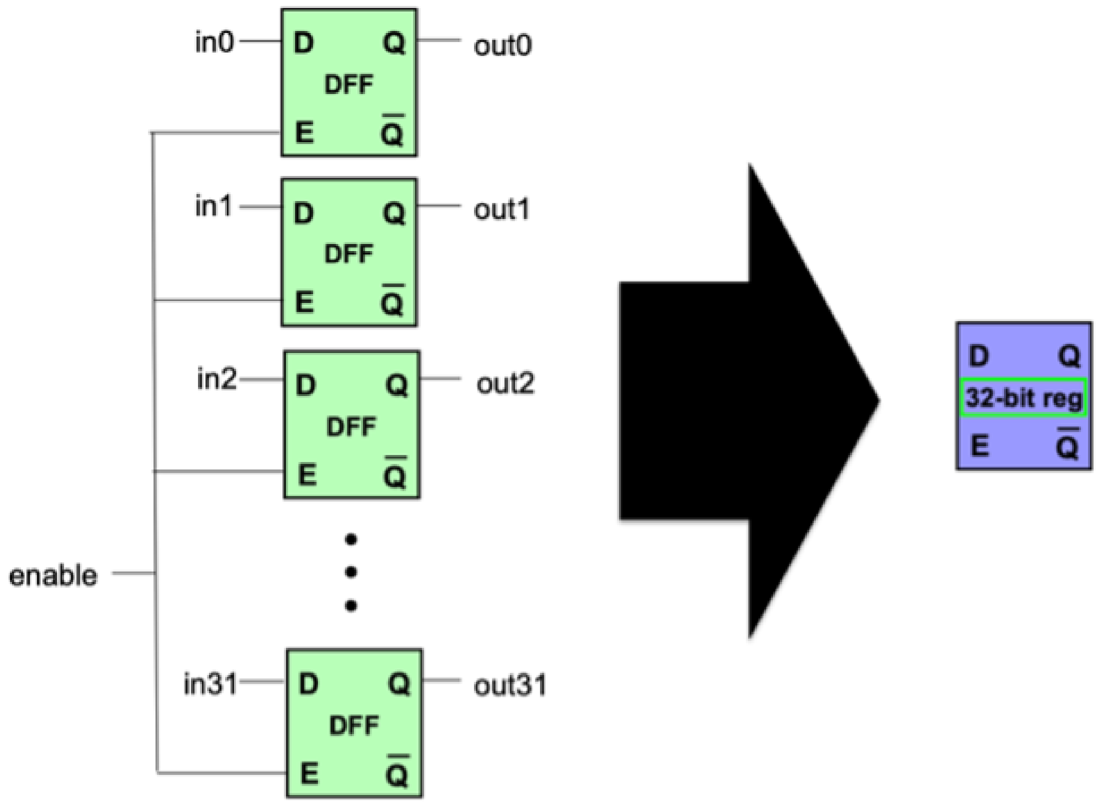
2.3. Write Port:



- The part marked "Write port" in the above diagram are the write port contained in the register file. First, I use decoder to convert ctrl_writeReg (5 bits) to 32 bits one hot. Second, I use AND gate to connect ctrl_writeEnable and each bit of the decoder result (ctrl_writeReg32bit). And then I can get the 32-bit output of my write_port module. This output can be used to decide which register to store data_writeReg in.
- Note: Register 0 must always read as 0 (no matter what is written to it, it will output 0). To realize this point, I let the data_writeReg of register 0

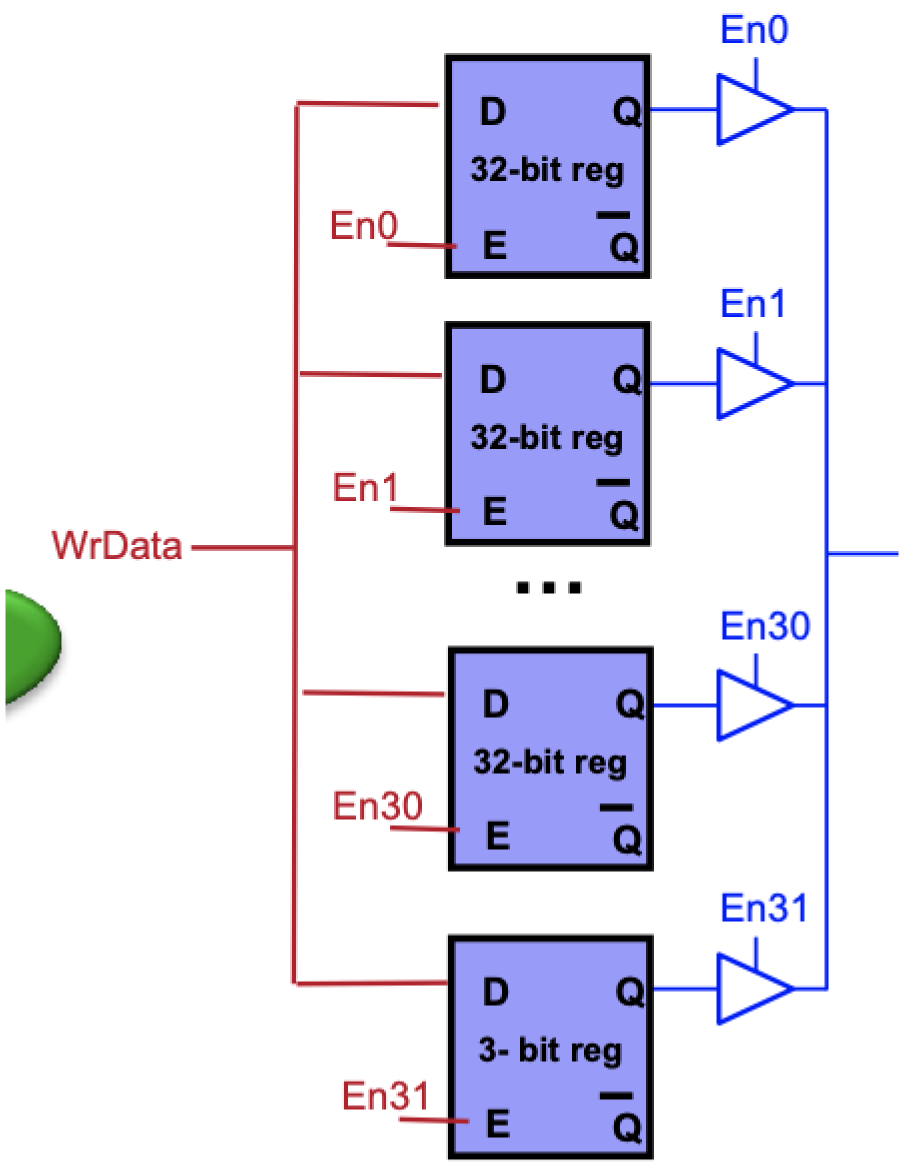
always be 0 (32-bit binary number).

2.4. 32-bit Register:



- Like the picture above, I will stick 32 DFFEs together to make a 32-bit register.

2.5. Register file:



- Like the picture above, I will stick 32 32-bit registers together to make a register file.