1 Introduction

For this lab, you are going to begin the construction of your simulated computer. The resulting component of this assignment is a 32×32 register file, that is a set of 32 registers each of which is 32 bits in size. See Figure 5.7 in your book for a graphical representation of the register file.

2 Requirements

The following requirements must be met by your register file:

1. The assignment will be written in VHDL. The exact signature of the register file should conform to the following definition:

```
ENTITY RegFile IS
    PORT (reg1, reg2, writeReg: IN STD_LOGIC_VECTOR(4 DOWNTO 0);
          WE, clock: IN STD_LOGIC;
          writeData: IN STD_LOGIC_VECTOR(31 DOWNTO 0);
          read1Data, read2Data: OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
END RegFile;
```

2. The register file will provide 2 outputs:

   Register 1 data is the data from one of registers as selected by the *read register 1* input. The data is 32 bits wide.

   Register 2 data is the data from one of registers as selected by the *read register 2* input. The data is 32 bits wide.

3. The register file will take 6 inputs:

   Read register 1 is the address within the register file of the register that should provide data for the *register 1 data* output. The address is 5 bits wide.

   Read register 2 is the address within the register file of the register that should provide data for the *register 2 data* output. NOTE: This address may be the same as the *read register 1* address. The address is 5 bits wide.

   Write data is the data that is sent to a particular register as selected by the *write register* input. The data is 32 bits wide.

   Write register is the address within the register file of the register that should received data from the *write data* input. The address is 5 bits wide.

   Write enable is a bit that indicates whether the register file should write the *write data* to the selected *write register*. A value of 1 indicates the register file should write, and a value of 0 indicates the register file should NOT write.

   Clock is the signal that controls when data is written to the *write register*.

4. All gates that are used must have four or fewer inputs.

5. Each gate should have a 5ps delay.

6. Register 0 should have a constant 0 value.

7. Memory should be implemented using a positive edge-triggered D flip-flop.

8. The D flip-flops should have a 20ps delay.

3 Implementation ideas

You will probably want to use two 32:1 multiplexors, a 5:32 decoder, and 31 registers as sub-components for your implementation. Creating those components will be much easier than trying to do all the work in a single entity.

Look into the VHDL `GENERATE` statement. You'll find it helps make your implementation more compact through less useless repetition.

The *write enable* input may not be stable. Do not count on it staying the same during a single cycle of the clock. The only guarantee you have is that it will hold the correct value at the positive edge of the clock.

You need to create a test bench that thoroughly exercises your register file. This test bench needs to include an implementation for the clock that the register uses. Make sure the period is long enough that you get stable values from the register file.

For this assignment, you should be able to build a structural model except for the D flip-flops (I've given you the behavioral model for the flip-flops). Try to keep to the structural model as it is good practice for "the real world."

4 Deliverables

You should turn in an electronic copy of your VHDL, including all the components you created to construct the register file hierarchically. Additionally, you must demonstrate your lab to me at a scheduled time.

The project is worth 100 points as follows:

- 25 points overall structure
  - 10 points good hierarchical design
  - 15 points correct connections between components
- 25 points multiplexor
  - 8 points supports 32 inputs
  - 5 points supports 32 bit wide data
  - 12 points correct functionality
- 30 points register
  - 5 points supports 32 bit wide data
  - 10 points correct functionality
  - 5 points register 0 remains constant 0
  - 5 points positive edge-triggered
  - 5 points glitches in *write enable* don't affect register
- 20 points decoder
  - 5 points supports 32 bit wide data
  - 5 points supports 32 outputs
  - 10 points correct functionality