Design of Digital Systems: Lab Assignment 2

The objective of this assignment is to design a register file, which is a unit that can hold a large amount of data.

Background

Computers need to be able to store and access information very quickly. However, simple digital systems like flip-flops, by themselves, cannot hold enough information to be useful. Thus register files are created that allow for one bus to access multiple locations, each holding multiple bits.

Program Specification

The design must meet the following specifications and use all components listed at least once. Some might be used multiple times.

Register

The register (name it `reg_module`) has the following features:

- Parameters:
  - `n`: The size of the word to store, in bits. To be called `n`.
- Inputs (One bit unless noted otherwise):
  - `clk`: Clock signal. Register file is rising edge triggered.
  - `rst`: Resets the register to all `0`s. Asynchronous active low.
  - `we`: Write enable. Enabling allows register contents to be written.
  - `din` (`n` bits): The content to be written
- Outputs:
  - `dout` (`n` bits): Reads the content stored in the register.

Hints

- Consider using the behavioral style for the register.

Multiplexer

Create an 8-to-1 MUX (name it `mux_8to1`), whose output and eight inputs are all `n` bits wide.

Decoder

Create a 3-to-8 decoder (name it `decoder_3to8`), with an enable signal to disable all output. The output of this decoder should not be written as eight separate signals, but instead as one 8-bit signal.

Register File

The register file (name it `reg_file`) contains eight registers each 4 bits wide and follows a one-read, one-write format. See Figure 1.

- Inputs (One bit unless noted otherwise):
  - `clk`: Clock signal.
  - `rst`: Resets all registers to `0`s. Asynchronous active low.
  - `we`: Write enable. Enabling allows register contents to be written.
  - `addr` (3 bits): Address of register to write to or read from.
  - `din` (4 bits): Data to be written to the selected register.
- Outputs:
  - `dout` (4 bits): Data to be read from the selected register.

Hints

- Notice that the read and write signals are 3 bits long, allowing them to address all eight registers.

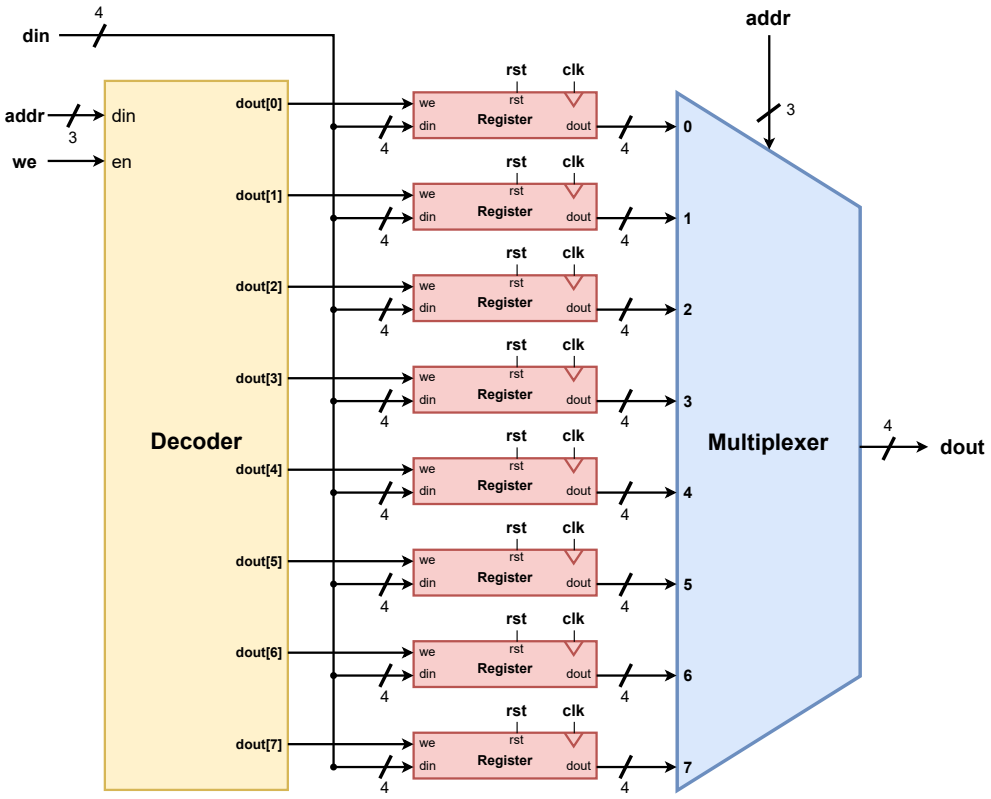

Figure 1: Register File

Testbench

To ensure that the register file is working properly, write a testbench that handles the following scenarios:

- Writing to each register
  - Write a random value to each register.
- Read the registers to check the previous written values are stored correctly
- Resetting the register file
  - The reset button should reset all registers to all `0`s.
  - Reread the value from each register to check that they are all `0`s

Report

- Do NOT collect timing results for this lab
- For the demo, test all the scenarios covered in the testbench

FPGA port map

IO	FPGA port
clk	E3
rst	Button CPU RESET
we	Button BTNL
addr	Switches 15-13
din	Switches 3-0
dout	LEDs 3-0