CMPE 260 Laboratory Exercise 2

Register File

Mohammed Fareed
Performed: February 22, 2024
Submitted: February 25, 2024

Lab Section: 4
Instructor: Prof. Richard Cliver
TA: Aubrey Tarmu
Henry Bang
William Tom

Lecture Section: 2
Professor: Prof. Marcin Lukowiak

By submitting this report, you attest that you neither have given nor have received any assistance (including writing, collecting data, plotting figures, tables or graphs, or using previous student reports as a reference), and you further acknowledge that giving or receiving such assistance will result in a failing grade for this course.

Your Signature: _____

Abstract

In this exercise, the design, implementation, and verification of a register file within a Field Programmable Gate Array (FPGA) environment were done, utilizing the Xilinx Vivado Design Suite. The primary focus was the creation of a digital system capable of storing and accessing multiple words, employing a two-read, one-write configuration. The project began with the specification of the register file, designed to support parallel data retrieval from two addresses, enhancing data access speed. Initial testing was conducted through behavioral simulation, ensuring the system's functionality met predefined specifications. This was followed by synthesis and post-implementation timing simulations, assessing the design's performance and identifying potential delays inherent to FPGA operations. The exercise also involved a hardware demonstration on a Basys3 FPGA platform. The results verified the register file's operational integrity and compliance with design requirements.

Design Methodology

The register module is the basic building block of our project, designed to store bits of digital information within the FPGA. A "2-D Array" structure was used for these modules to organize data storage and access. The number and size of the registers was parameterized, using the following constants in a global package:

- `BIT_DEPTH`: The size of the data bus (register size).
- `LOG_PORT_DEPTH`: The address bus size ($\log_2$(number of registers)).

The register file combines these modules into a system that allows for parallel data access, adopting a two-read, one-write approach. This setup significantly improves the efficiency of data retrieval, reducing the time required for retrieving instruction operands and enhancing the system's overall performance.
The system's operation is controlled by various input signals:

- `clk_n`: Falling edge-triggered clock signal.
- `we`: Write enable signal.
- `Addr1`, `Addr2` (`LOG_PORT_DEPTH` wide): Address lines for reading data.
- `Addr3` (`LOG_PORT_DEPTH` wide): Address line for writing data.
- `wd` (`BIT_DEPTH` wide): Data to be written to the register file.

The `clk_n` signal, ensures that data writing and reading are synchronized. The write enable (`we`) signal allows data to be written to the registers, protecting against accidental data loss. Two address lines (`Addr1` and `Addr2`) are used for reading data, while a third address line (`Addr3`) is used for writing data. The data to be written is specified by the `wd` signal. The register file has the following outputs:

- `rd1`, `rd2` (`BIT_DEPTH` wide): Data read from the register file.

The `rd1` and `rd2` signals output the data read from the register file at the addresses specified by `Addr1` and `Addr2`, respectively.

A specific feature of the design is making register 0 immutable, meaning it always outputs zero. This decision ensures a constant zero value is available for computations requiring it, thus adding to the system's functionality.

Results and Analysis

The register file was implemented using VHDL, with the global constants defining the register size and address bus size to be 32 bits and 5 bits, respectively. The design was tested using behavioral simulations, ensuring the system's correct operation. Synthesis and post-implementation timing simulations were then performed to assess the design's performance of FPGA operations. The following are all the test cases that were run on the register file:

we	Addr1	Addr2	Addr3	wd	RD1	RD2
'0'	000	000	001	10	00	00
'1'	000	000	001	10	00	00
'1'	001	000	010	ff	10	00
'0'	010	011	100	20	ff	00
'1'	011	100	101	33	00	00
'0'	100	101	110	44	00	33
'1'	101	110	111	55	33	00
'1'	110	111	000	66	00	55
'1'	111	000	001	77	55	00
'0'	000	001	010	88	00	77

Table 1: Simulation Test Cases.

The test cases in Table 1 were run on the register file, with assert statements to ensure the correct operation of the design. The test cases mainly focus on testing the operation of the `we` signal, correctly reading data from the register file after writing to it, and ensuring that register 0 can't be written to, always returning zero.

The results of the simulations were observed to ensure the register file's correct operation and timing. Figure 1 shows the behavioral simulation results of the register file.

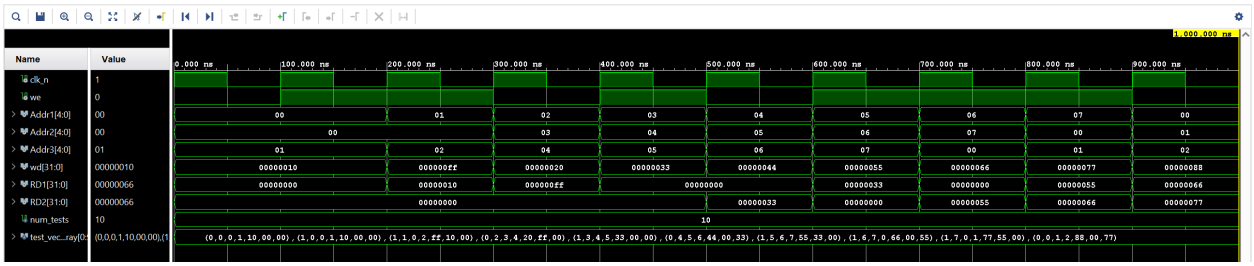

Figure 1: Behavioral Simulation Results.

The figure above shows the behavioral simulation results of the register file, with the input and output signals labeled. The simulation results demonstrate the correct operation of the register file, with the read data matching the written data. The simulation results also show the read data being zero when the address is zero, as expected. It also shows th e data `0xff` to address 2, then reading from address 2, which correctly returns `0xff`.

The register file was then implemented on a Basys3 FPGA, and the design was verified through a post-implementation timing simulation. Figure 2 shows the post-implementation timing simulation results of the register file.

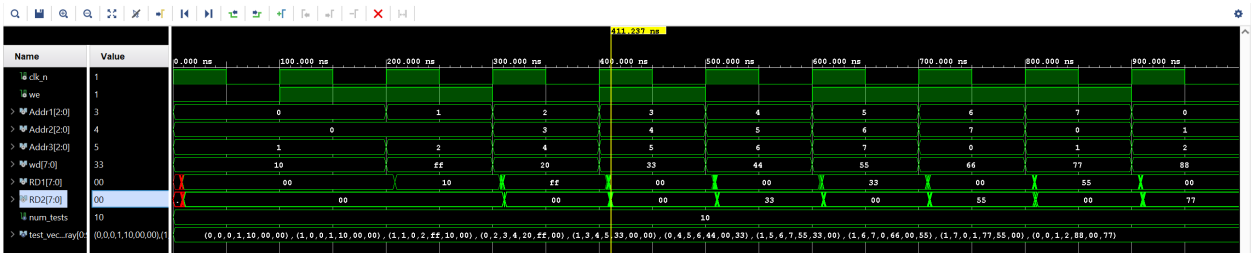

Figure 2: Post-Implementation Timing Simulation Results.

The figure above shows the post-implementation timing simulation results of the register file. The simulation results demonstrate the correct operation of the register file, with delays within the expected range, which ensures the design's performance and timing. An average delay of around 11 ns was observed on `rd1` and `rd2` signals, which is within the acceptable range for the design.
The results of the hardware demonstration on the Basys3 FPGA platform verified the register file's operational integrity and compliance with design requirements.

Conclusion

The design, implementation, and verification of a register file within a Field Programmable Gate Array (FPGA) environment were successfully completed. The register file was designed to support parallel data retrieval from two addresses, enhancing data access speed. The design was tested using behavioral simulations, ensuring the system's correct operation. Synthesis and post-implementation timing simulations were then performed to assess the

design's performance of FPGA operations. The results of the simulations demonstrated the correct operation of the register file, with delays around 11 ns, which are within the expected range, ensuring the design's performance and timing. The hardware demonstration on the Basys3 FPGA platform verified the register file's operational integrity and compliance with design requirements.

Exercise 2: Register File

Student's Name: Mohammed Fareid Section: 4

PreLab		Point Value	Points Earned	Comments
PreLab	MIPs Questions	10	10	agt 2-8
	Testbench Cases	5	5	

Demo		Point Value	Points Earned	Date
Demo	Behavioral Simulation	10	10	agt 2-22-24
	Post-Synthesis Timing Simulation	5	5	
	Synthesis Schematic	5	5	
	Post-Implementation Timing Simulation	5	5	
	Implementation Utilization Report	5	5	
	EDAPlayground Behavioral Waveform	5	5	
	Hardware Demonstration	10	10	

To receive any grading credit students must earn points for both the demonstration and the report. Students must also submit all VHDL code as .TXT files. Failing to do so may result in no credit for the assignment.