COEN 316: COMPUTER ARCHITECTURE & DESIGN

Section DM-X

LAB REPORT # 2: ARITHMETIC & LOGIC UNIT

INSTRUCTOR:
Dr. Fadi Alzhouri

Lab Coordinator:
Ted Obuchowicz

Submitted by:
Jaskirat Kaur
40138320

"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"

Date Written: Wednesday, October 18, 2023

Objectives
Design a 32 bit register file using VHDL and simulate the design using ModelSim.
This register will then be used in subsequent labs as part of component of a CPU. There is a given entity specification to be used for the design of the register.

Introduction
The CPU's register is a crucial component, as it stores data that the Arithmetic Logic Unit (ALU) will work on. This data can be various things, like instructions or memory addresses. The lab consisted of designing a 32 bit register file using VHDL. The Register file contains 32 registers of 32 bits each. It is driven by a clock signal, where writing to the registers only occurs on a
rising clock edge. There are various control signals to read data and select registers.

Theory

The register file contains 32 registers, each with 32 bits. You can reset the entire register file by using a special reset signal.

Reading Values:

There are two places where you can read the contents of these registers, and you decide which registers to read by providing a 5-bit address for each of these reading places. The reading happens independently of the clock input, which means it's not tied to a specific timing.

Writing values:

The register file has a single input port called "din," where you can send 32-bit data to be placed into a particular register. To decide which of the 32 registers should receive this data, you provide a 5-bit address through the "write_address" input. The actual writing of this data into the chosen register occurs in sync with a rising clock edge. This writing process is also controlled by the "write" signal, which needs to be active (set to '1') and coincide with the transition of the clock input from '0' to '1' for the write operation to take place.

To design the register file using VHDL, processes will be used. To simulate the code using Modelsim simulator, a do file with the adequate inputs will verify all the possible operations of the register file. Finally, an xdc file will be used to implement the register design on a Nexys A7 FPGA board.

Results:

Implemented Design:

Elaborated design:

Do file:

The 32-bit register code simulated correctly. And the Board-wrapper version of the register was implemented in vivado. The waveform above shows possible input values that were tested.

Conclusion

To conclude, in this second lab, the 32-bit register was coded and simulated using processes and CSA statements VHDL. It was then implemented on the Nexys board using the software Vivado. It is now to be seen how this register will be used as a component of the CPU that will be build in the subsequent labs.

Appendix:

Reg.vhd:

Reg_board.vhd:

Xdc.xdc file: