Virginia Commonwealth University
College of Engineering

LAB 3

"On my honor, I have neither given nor received aid on this assignment, and I pledge that I am in compliance with the VCU Honor System."

Introduction

This lab, I designed a 32-bit arithmetic logic unit ALU in VHDL. I implemented ALU needs to support, at a minimum, the following instructions:

- Addition
- Subtraction
- Bitwise AND
- Bitwise OR
- Logical left shift
- Logical right shift
- Arithmetic left shift
- Arithmetic right shift

In addition to the instruction output, the ALU must output a zero bit, overflow bit, and carryout bit. I implemented my ALU using a modular design. By use of the generate statement I created single bit operators and cascaded them across the 32 bit data bus. In addition to the I created a register file 32-bit register file. Instead of using an array of logic vectors, I choose to uses 32 individual signals so when synthesized the design will still work without modifications or creating delays. I referenced schematic from the lab assignment.

Design

Add Sub

My file structure is as follows, for adding and subtracting, I implemented the full bit adder and subtractor. Where the data2 signal is xor with the mode if the mode is 0 data2 remains the same, if mode is 1 data2 is inverted (ones complement). The first carry in for the addition will be the mode. If mode is 0 this indicates addition will take place, if mode is 1 this indicates subtraction. When the first carry in is 1, the ones complement of data2 becomes the 2s complemented, allowing me to use the full adder to preform subtraction without any modifications to the hardware.

To get the carryout bit I created an internal signal of 33 bits which kept track of the resulting carryout for each individual bit operation. The MSB of the interval carryout signal was sent to the carryout flag.

To get the overflow flag bit I used the internal signal of 33 bits which kept track of the resulting carryout for each individual bit operation. The MSB of the internal signal and the bit before are xor together. This logic can be used for both addition and subtraction.

To get the zero flag bit I created an internal signal of 32 bits which kept track of the result of each bits output. This buffer is needed because you cannot use an output signal whatsoever. I checked each bit of this buffer and checked if all bits where zero.

AND

This was simple I created an and 2 bit operator and used the generate to generate 32 times.

OR

This was simple I created an and 2 bit operator and used the generate to generate 32 times.

Shifting

I created an internal signal that was 33 bits, and moved the input one index to the left or one to the right depending on the type of shift. Then I truncated the 33 bit signal to 32 bits. Athematic shift left, is the same as logical shift left. Arithmetic shift right requires that the sign stats the same. So, for this I made a conditional statement that extends the MSB to the 33 bit and the 32 bit of the internal signal. Then truncate the internal signal.

Alu

The alu is used to string all the components together. All the components gets the data from the incoming data bus, but the mux is respirable for choosing which output is correct.

Alu mux

This was responsible decoding the ALUctr and determine which output to the result of the ALU.

Register file

As mention earlier, I chose to have 32 individual signals that and array of 32 vectors. I feel the 32 signals reflects how the hardware actual works where the 32 array is a more programing point of view.

Problems

The main problems were thinking of this project in the point of view of programming. Where VHDL is a description of hardware. This fact has led me down several dead ends. Often, I would try to solve coordination or logical operations by creating a new process which through off the entire ALU. Having the wrong triggers in process sensitive list.

The other problem was having the correct output delayed when a new instruction was sent. I solved this problem by created a fast-internal clock that would trigger the processes to check if a new output was needed.

Conclusion

I feel about of my time was wasted on thinking as a programmer instead of designing hardware. While I started early, because my perspective was incorrect I wasted time. Once I understood that this is hardware not software I made more and more progress.

Also I have been using git to save my work, and the gitignore file helps ensure that I am not flooded with random extra files. It has helped me greatly in keeping track, also working from any machine.

Appendix

SIMULATION IN SIMULATION PDF