CS2300

VERILOG LAB 4
CPU DESIGN

Introduction

I have made a CPU in Verilog. The CPU consists of a Control Unit (CU) and an Arithmetic and Logical Unit (ALU).

The CU takes an operation code (opcode) as input. This opcode specifies 1 or 2 operands and 1 operation. The CPU then performs this operation on the operand(s) and returns the answer.

My CPU can perform 7 different operations: Addition, Subtraction, Increment, Decrement, Bitwise AND, OR, NOT.

I did this task using only structural descriptions of the modules. I also tried to use the concept of abstraction to a good degree.

The Control Unit

The CU takes an operation code (opcode) as input and gives the operation to be performed and the operands as output. I did this by picking up the respective bits from the opcode and assigning these to the corresponding “wire” buses.

The Arithmetic and Logical Unit

The CU takes an operation(3 bit) and 2 operands (8bit) as an input.

It performs all 7 operations and selects exactly one of these to be returned as output.

The selection is achieved using an 8x1 multiplexer which takes the 3 bit operation as the select line.

The 3-bit codes for the 7 operations:

ADD - 001

SUBTRACT - 010

INCREMENT - 011

DECREMENT - 100

AND - 101

OR - 110

NOT - 111

Fundamental Modules

And, Or, Not, Xor

I used the predefined modules for these operations.

Half Adder, Half Subtractor, Full Adder

I implemented the half adder and subtractor modules using the fundamental gates.

The full adder module is made using 2 half adders.

Modules used in ALU

Adder

An 8 bit adder is implemented using an array of full adders where the carry is rippled through the adders [ripple carry adder].

Subtractor

An 8 bit subtractor is implemented by

1) taking the 1's complement of operand 2
2) Adding this complement to operand 1 along with an input carry 1

The 1's complement of operand 2 with the input carry act like the 2's complement of operand2. Thus, adding this to operand 1 gives the answer to op1-op2 in 2's complement form.

Incrementer

An 8 bit incrementer is implemented using an array of half adders. Increment is the same as adding 00000001 to the number. We do not require a full adder because at any point, only 2 potential non-zero numbers are added.

For LSB: the LSB of operand is added to 1

For the rest of the bits: the carry from the previous bit is added to the current bit.

Decrementer

The same logic of incrementer is used, but with half subtractors.

Bitwise AND, OR, NOT

These are implemented using arrays of the corresponding fundamental gates.

Multiplexer

2x1 multiplexer

This is implemented using AND and NOT gates.

It takes one select bit and 2 buses as input, and returns 1 bus as an output.

One of the input buses is selected in accordance with the select bit.

4x1 multiplexer

This is implemented using 3 2x1 multiplexers.

It takes 2 select bits and 4 buses as input, and returns 1 bus as an output.

The 2nd select bit is used to select between (input0 and input1) and (input2, input3).

Then the 1st select bit is used to select between the results of the above 2.

8x1 multiplexer

This is implemented using 2 4x1 multiplexers and 1 2x1 multiplexer.

It takes 3 select bits and 8 buses as input, and returns 1 bus as an output.

The 2nd and 1st select bits are given to the 4x1 multiplexers.

Then the 1st select bit is used to select between the results of the above 2.

Waveforms

The instruction, when written in hexadecimal, has 3 digits

The MSD corresponds to the operation, the next 2 to operand1 and the last 2 to operand 2 [both in hexadecimal]