COMP4300

Lab Exercise Two

Objective

This lab develops the first building block for this semester's CPU, the arithmetic-logic unit (ALU). It will be combined with the other data path components from later labs, resulting in a complete CPU by Lab 4.

Instructions

Develop VHDL for the following component. You should define an architecture for the ALU entity given below. You should test your architecture by developing simulation files for the entity. Your architecture should implement the functionality described in this document. To make the simulation results more readable, we will use a 32-bit datapath.

You should use the types from the package "dlx_types" (available on the files section of Canvas) and the conversion and math routines from the package "bv_arithmetic" in files bva.vhd, and bva-b.vhd on Canvas. The propagation delay through the ALU should be 15 nanoseconds. Code to make use of them should be included in your vhdl file for the ALU:

```
use work.dlx_types.all;
use work.bv_arithmetic.all;
```

Arithmetic-Logic Unit This unit takes in two 32-bit values, and a 4-bit operation code that specifies which ALU operation (eg. add, subtract, multiply, etc) is to be performed on the two operands. For non commutative operations like subtract or divide, operand1 always goes on the left of the operator and operand2 on the right.

The operation codes are

0000 = unsigned add

0001 = unsigned subtract

0010 = two's complement add

0011 = two's complement subtract

0100 = two's complement multiply

0101 = two's complement divide

0111 = bitwise AND

1001 = bitwise OR

1011 = bitwise NOT of operand1 (ignore operand2)

1100 = pass operand1 through to the output

1101 = pass operand2 through to the output

1110 = output all zero's

1111 = output all one's

The unit returns the 32-bit result of the operation and a 4-bit error code. The meaning of the error code should be

0000 = no error

0001 = overflow (either negative or positive)

0010 = divide by zero

The ALU should be sensitive to changes on all the input ports. Note that the ALU is purely combinational logic, no memory, and that there is no clock signal.

The entity declaration is

```
entity alu is

generic(prop_delay : Time := 5 ns);

port(operand1, operand2: in dlx_word; operation: in

          alu_operation_code;

          result: out dlx_word; error: out error_code);

end entity alu;
```

Deliverables

Please turn in the following things for this lab:

1. The text of your VHDL source code.

2. Your simulation test file. Do not exhaustively test these designs since they take lots of input bits, but do test a reasonable number of things. For example, test both normal function and an error, if applicable (e.g. overflow for unsigned add, subtract, multiply, nothing for and/or/not) for each of the alu operations.

3. Transcripts of tests or screenshots running your simulations. It should be clear what is being tested. If not, add text to explain it. You cannot test exhaustively but you should test each function code for correct operation and demonstrate how each error code is generated.

Please turn in all files on Canvas. If I have questions, I may ask you to schedule a time to demo your code, if I can't figure out how something works by reading the code.