COMP4300

Lab Exercise Two

Objective

This lab develops the first building block for the MIPS datapath, the ALU. It will be combined with the other data path components from later labs, resulting in a complete MIPS by Lab 4.

Instructions

Develop VHDL for the following MIPS 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 (both addr and data buses), not 64 bit.

You should use the types from the package "dlx_types" (available on the files section of Canvas) and the conversion routines from the package "bv_arithmetic" in files bva.vhd, and bva-b.vhd on Canvas. The propagation delay through the ALU should be 5 nanoseconds for any operation.

We will only implement a subset of the MIPS ALU operations. We will not implement any floating point operations. We will not implement any byte or half-word operations. We will only implement the following 32-bit ALU operations.

```
ADD
ADDU
SUB
SUBU
MUL
MULU
AND
OR
```

SLT

However, the ALU should take a 4-bit operation input that tells which of sixteen functions to implement. The codes to use are

ADD = 0 (used for ADD, ADDU, ADDI, ADDIU instructions)

SUB = 1 (used for SUB, SUBI, SUBU, SUBIU instructions)

AND = 2 (used for AND, ANDI instructions)

OR = 3 (used for OR instructions)

XOR = 4 (not used)

unused = 5 (not used)

SLL = 6 (unused)

SRL = 7 (unused)

SRA = 8 (unused)

SEQ = 9 (unused)

SNE = a (unused)

SLT = b (used for SLT)

SGT = c (unused)

unused = d

MUL = e (used for MUL, MULU)

DIV = f (unused)

Note that the unsigned and signed versions of an operation have the same function code.

For the codes that I haven't asked you to implement, return zero as the result of the ALU operation (or for 10 points extra credit, implement all sixteen ALU operations). Googling will tell you semantics (what the instruction does) of each MIPS ALU operation.

There is an additional one-bit input "signed" that is set to 1 if the operation takes signed (32-bit two's complement) inputs and set to 0 if the operation takes unsigned (32-bit positive binary integer) inputs. If the operation makes no sense when the inputs are signed (i.e. SRL, SLL, AND, OR, XOR), the signed bit is ignored. The ALU also returns a 4-bit condition code, of which the only conditions we will use are

0000 = no error

0001 = over/underflow

The entity declaration should look like:

```
entity alu is

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

        signed: in bit;

        result: out dlx_word; error: out error_code);

end entity alu;
```

The ALU should be sensitive to changes on all the input variables.

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, for the ALU, be sure to test every function, and for those that generate error codes, test the error conditions.

3. Transcripts of tests or screenshots running your simulations. It should be clear what is being tested. If not, add text to explain it.

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.