COMP4300

Lab Exercise Three

Objective

This lab develops some remaining datapath building blocks for the Aubie processor . It will be combined with the Aubie control logic to make a working cpu in Lab 4.

Instructions

Develop VHDL for the following components. You should define an architecture for each of the entities given below. You should test each entity by developing simulation files for the entity. Your architecture should implement the functionality described in the text for each entity.

You should use the types from the dlx_types and bv_arithmetic packages you used in lab2.

32-bit single-value register. This will be used everywhere in the chip that a temporary value should be stored. The propagation delay for the unit should be 10 ns.

```
entity dlx_register is
     port(in_val: in dlx_word; clock: in bit; out_val: out
dlx_word);
end entity dlx_register;
```

The register should be sensitive to all inputs. If `clock` is one, the value present at `in_val` should be copied to `out_val` . When `clock` goes to zero, the output value is frozen until `clock` goes high again.

Register File

This is the unit where there numbered registers R0-31 are found. The propagation delay through the register file should be 15 nanoseconds for a read operation (zero for write, but write has no output). The reg_number is a five-bit number which specifies which register is being read or written. The register file can do one read or one write per clock cycle. If a read is being done (readnotwrite is 1), the data_in input is ignored, and the value in register reg_number is copied to the data_out port. If a write is being done (readnotwrite is 0), the value present on data_in is copied into register number reg_number. The data_out port does not have a meaningful value for a write.

The entity declaration should look like:

```
entity reg_file is
  port(data_in : in dlx_word;  readnotwrite, clock: in bit;  data_out: out
     dlx_word; reg_number : in  register_index);
```

```
end entity reg_file;
```

The entity should be implemented with an architecture consisting of a single VHDL process. You should use an array variable of 32 dlx_words to store the register values, something like

```
type reg_type is array (0 to 31) of dlx_word;
…
variable registers  : reg_type;
```

There are two kinds of multiplexer: two-way and three-way, depending on how many inputs are present. The multiplexer copies the input named like the value of the which input to the output (that is, if which = 0, copy input_0 to the output, etc)

Two-way multiplexer

```
entity mux is
     generic(prop_delay : Time := 5 ns);
     port (input_1,input_0 : in dlx_word; which: in bit; output: out dlx_word);
end entity mux;
```

Three-way multiplexer

```
entity threeway_mux is
     generic(prop_delay : Time := 5 ns);
     port (input_2,input_1,input_0 : in dlx_word; which: in threeway_muxcode;
output: out dlx_word);
end entity threeway_mux;
```

PC Incrementer

This unit increments the 32-bit unsigned value at its input port when clock transitions to one. Don't worry about behavior when it overflows; it can just go back to zero.

```
entity pcplusone is
   generic(prop_delay: Time := 5 ns);
   port (input: in dlx_word; clock: in bit;  output: out dlx_word);
end entity pcplusone;
```

Deliverables

Please turn in the following things for this lab:

- A printout of your VHDL code.
- 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.

- Transcripts/screenshots of tests running your simulations. You cannot test exhaustively, but you should demonstrate that all your modules work.
-

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.