ECE 238L
Digital Computers and Number Systems
Lecture lab 7

November 2, 2006

Problem – Programming VHDL can be tedious and error prone

- There other behavioral and structural circuits.
- VHDL has some structures that are very close to structural, but more convenient.
- I prefer using the when/else construct and will provide examples.
- Continue to use the provided flipflops, but these other structures will provide us some easier middle ground. (Don't go and write funky processes. I won't help you and you will loose points on your code.)

Example – Arrays of Vectors

```
type reg_array_t is array(7 downto 0) of
                                std_logic_vector(15 downto 0);

signal registers : reg_array_t;
signal reg_sel : std_logic_vector(7 downto 0);

-- enumerated types - may use in statemachines...
type opcodes is (add, sub, jump, call);  -- Type with 4 values
signal instruc: opcodes;                 -- Signal of this type
```

Example – Decoder 3 to 8

```
-- 3 to 8 decoder
reg_sel <= "00000001" when dr = "000" else
           "00000010" when dr = "001" else
           "00000100" when dr = "010" else
           "00001000" when dr = "011" else
           "00010000" when dr = "100" else
           "00100000" when dr = "101" else
           "01000000" when dr = "110" else
           "10000000" when dr = "111" else
           "00000000";
```

Example – Mux 128 to 16 bit

```
ra <= registers(0) when sr_a = "000" else
      registers(1) when sr_a = "001" else
      registers(2) when sr_a = "010" else
      registers(3) when sr_a = "011" else
      registers(4) when sr_a = "100" else
      registers(5) when sr_a = "101" else
      registers(6) when sr_a = "110" else
      registers(7) when sr_a = "111" else
      x"0000";
```

Example – for/generate

```
-- registers
 reg_gen: for i in 7 downto 0 generate
   begin
     reg_inst : component dff16_ce
       port map (
         d => input,
         clk => clk,
         clr => reset,
         en => regWE,
         q => registers(i));
   end generate reg_gen;
```

Example – tristate

```
buss <= my_16bit_wire when ctrl = '1' else
      (others => 'Z');
```

This is not the only way, but this is a straight forward way to do this. **others** is an aggregate. There are other uses for it as well:

```
address <= (31 => '1',30 => '0', others => '1');
bus <= ( 4 ='1', others => '0');
```

Example – Addition and other operators

```
add <= a + b; -- this can be supported (and is for FPGAs)
sub <= c - d; -- this, too.
mul <= e * f; -- this is no good, because there are many options.
div <= g / h; -- this no worky
              -- unless h is a constant that is 2^n.

div2 <= g / conv_to_std_logic_vector(2, 16); -- which is :

div2 <= '0' & g(15 downto 1);

shf <= a sla b; -- the shifts don't work either.
```

The set of operations that work really depend on the technology you are targeting (and the synthesizer.) If you aren't careful you may not get what you want (even if you get some result out.)