Using VHDL to Design Digital Circuits – Part 1

- assignments
- processes and if-then-else
- avoiding unintended storage
- separation principle

Hardware Description Languages

- HDLs allow designers to work at a higher level of abstraction than logic gates
- As with programming languages, HDL descriptions are compiled into a lower level representation
  - low level form can be simulated for logical correctness
  - and, can be converted to a circuit specification using a library of primitive components and a set of constraints that affect circuit cost and performance
- But don’t confuse hardware design with software
  - software is translated into sequential instruction sequence
  - HDL descriptions must translate to physical circuits
  - circuits are inherently parallel with many things going on at once

Binary Coded Decimal to Excess-3 Code

- Binary Coded Decimal to Excess 3 converter
- ABCD is 4 bit input value
- WXYZ is 4 bit output value
- WXYZ = ABCD + 3

Vector Assignments

- Binary Coded Decimal to Excess 3 converter
- \( x3 \leftarrow bcd + 3 \)

Conditional Signal Assignment

- The conditional signal assignment can make logic equations easier to understand (and write)
- Example:
  \[ c \leftarrow x \text{ when } a \neq b \text{ else } y \text{ when } a = '1' \text{ else } z; \]
- General form:
  \[ x \leftarrow \begin{cases} v_1 \text{ when } \text{condition}_1 \text{ else } \vdots \text{ else } \vdots \text{ else } v_n \end{cases} \]
Practice Questions

1. Draw a circuit that implements each of the following VHDL code segments, assuming $a$-$d$ are all single bit signals in both cases:

   \[ x \leftarrow a \text{ and } b; \]
   \[ y \leftarrow a \text{ or } c \text{ and } d; \]
   \[ z \leftarrow (a \text{ and } c) \text{ or } (x \text{ or } c); \]

2. Write a VHDL code segment that implements the circuit shown below:

```vhdl
x <= a and b;
y <= a or c and d;
z <= '111';
```

Practice Questions

1. Draw a circuit that implements each of the following VHDL code segments, assuming $a$-$d$ are all single bit signals in both cases:

   \[ x \leftarrow a \text{ and } b; \]
   \[ y \leftarrow a \text{ or } c \text{ and } d; \]
   \[ z \leftarrow (a \text{ and } c) \text{ or } (x \text{ or } c); \]

2. Write a VHDL code segment that implements the circuit shown below:

```vhdl
x <= (A and B) \text{ when } c = '0';
\quad \text{when } c = '1';
\end{align*}
```

Selected Signal Assignment

- Selected signal assignment can be viewed as special case of conditional signal assignment.

- Example:

```vhdl
with x select
  c <= '0' when '00',
       '1' when '01' \text{ or } '10',
       \text{else};
```

- can be implemented using a single multiplexor stage.

- Resulting circuit is more compact and faster than circuit produced by conditional signal assignment.

Comments in VHDL

- Add a comment to a line of VHDL, using two dashes; e.g., \textit{this is a comment}.

- Every source code file should start with a short comment explaining its role.
  
  - good idea to include your name and date, as well

- Every VHDL module (entity-architecture pair) should be preceded by a comment.
  
  - explain what the module does.

- Comment signal, function, procedure declarations.

- Comment major sections of architecture.
  
  - help reader understand purpose of different code sections.
  
  - don't just re-state the obvious.
Processes and if-then-else

- Example:
  ```vhdl
  entity foo is port(
    a, b: in std_logic;
    c, d: out std_logic_vector(3 downto 0));
  end foo;
  architecture bar of foo is begin
    process (a, b) begin
      if a /= b then
        c <= "0010";
        d <= "1100";
      elsif a = '1' then
        c <= "1101";
        d <= a & b & "01";
      else
        c <= "0100";
        d <= "10" & b & a;
      end if;
    end process;
  end bar;
  ```

- Sensitivity list for combinational circuits must include all signals whose values are used in process.

“Conflicting” Assignments

- The code segment:
  ```vhdl
  architecture bar of foo is begin
    a <= '1'; b <= a; a <= '0';
  end bar;
  ```
  is incorrect, because the two assignments to `a` conflict.

- However, the following is allowed:
  ```vhdl
  architecture bar of foo is begin
    process(a) begin
      a <= '1'; b <= a;
      a <= '0';  -- note: b = '0'
    end process;
  end bar;
  ```

- In such situations, the first assignment is ignored.

- Within process, assignments that come later in text, logically replace earlier assignments to same signal.

Avoiding Unintended Storage

- Within a process, if value of a signal is not specified for some input condition, it means that signal is unchanged.

- Example:
  ```vhdl
  process(a, b) begin
    if a = '1' then
      x <= '0';
    elsif b = '1' then
      x <= '1';
    end if;  -- x retains its value when a=b=0
  end process;
  ```

- Storage elements are required to implement circuit with the specified behavior.
  - If one accidentally omits a condition for a signal, unintended storage elements are synthesized.

- Easy way to avoid unintended storage is to start process with assignment of default values to all signals assigned a value inside the process.

Default Values

- Example:
  ```vhdl
  entity foo is port(
    a, b: in std_logic;
    c, d: out std_logic_vector(3 downto 0));
  end foo;
  architecture bar of foo is begin
    process (a, b) begin
      c <= "0100";
      d <= "10" & b & a;
    end if;
    end process;
  end bar;
  ```

- Initial assignments define “default” values for `c` and `d`.

Separation Principle

- A VHDL code segment defining several signals can be re-written to separate the different signals.

  ```vhdl
  x <= x"0000";
  y <= x"abcd";
  if a = b then
    x <= y;
  elsif a > c then
    y <= b;
  else
    x <= x + y;
  end if;
  ```

- Code segment defining `x`:

  ```vhdl
  y <= x"abcd";
  if a = b then
    x <= y;
  elsif a > c then
    y <= b;
  else
    x <= x + y;
  end if;
  ```

- Statement order matters within each segment, but order of segments does not matter.

Exercises

1. Rewrite the following VHDL module using only ordinary signal assignments (no conditional assignments):

   ```vhdl
   entity foo is port(
     a, b, c, d: in std_logic;
     x, y: out std_logic_vector(3 downto 0));
   end foo;
   architecture arch of foo begin
     x <= b and a when a='0' else not c when a='1' else a or d;
     y <= (a and d) when a='1' else (b or not c) when c='0' else c when others;
   end arch;
   ```
Solutions

1. The revised architecture is:
   architecture arch of foo begin
   x <= ((not a) and b and c) or (a and (not b) and (not c)) or (a and b);
   y(1) <= a or b or (not c);
   y(0) <= (a and d) or ((not a) and b and c);
   end arch;

2. The body of the revised architecture appears below:
   begin
   z <= b xor c;
   process (a,b,c,d,z) begin
   if a = "000" then x <= z; y <= z;
   elsif a = "001" or a = "100" then x <= c; y <= c;
   elsif a = "101" and a <= "111" then x <= d; y <= d;
   elsif a = "011" then x <= b+c; y <= b+c;
   else x <= c+d; y <= c+d;
   end if;
   end process;
   end arch;