FPGA Design
EECE 277

Introducing VHDL Constructs

Dr. William H. Robinson
February 9, 2005

http://eecs.vanderbilt.edu/courses/eece277/

Topics

“It’s great to learn, ‘Cause knowledge is power!”
– Schoolhouse Rock

• Administrative stuff
  – Homework Assignment #3 (due Monday, February 14)
  – Remember, Exam I is next Wednesday
• Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (HDL)
• VHDL for simulation
• VHDL for synthesis

VHDL for Digital Logic

• VHDL is a language used for simulation and synthesis of digital logic

• Simulation
  – Executes the VHDL model to mimic behavior
  – Answer questions such as design correctness or performance
  – Must trade off accuracy vs. simulation speed

• Synthesis
  – VHDL model is input to synthesis compiler
  – Generate physical circuit design

Example: Majority Function

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Majority</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

What is the Boolean Function?

Hint: you may want to use a Karnaugh map
Example: Majority Function

Corresponds to Schematic Design Entry

VHDL Description

library ieee;
use ieee.std_logic_1164.all;

entity major_fun is
  port (A, B, C : in std_logic;
        Y : out std_logic);
end major_fun;

architecture a1 of major_fun is
begin
Y <= (A and B) or (A and C) or (B and C);
end a1;

Example: Majority Function

VHDL Description

library ieee;
use ieee.std_logic_1164.all;

entity major_fun is
  port (A, B, C : in std_logic;
        Y : out std_logic);
end major_fun;

architecture a1 of major_fun is
begin
Y <= (A and B) or (A and C) or (B and C);
end a1;

VHDL Design Entity

- Interface
  - Connections to the system

- Behavior
  - Type of processing on input signals and type of output signals produced
### Entity Ports

<table>
<thead>
<tr>
<th>Mode</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>Used for a signal that is an input to an entity.</td>
</tr>
<tr>
<td>OUT</td>
<td>Used for a signal that is an output from an entity. The value of the</td>
</tr>
<tr>
<td></td>
<td>signal cannot be used inside the entity. This means that in an assignment</td>
</tr>
<tr>
<td></td>
<td>statement, the signal can appear only to the left of the &lt;= operator.</td>
</tr>
<tr>
<td>INOUT</td>
<td>Used for a signal that is both an input to an entity and an output</td>
</tr>
<tr>
<td></td>
<td>from the entity.</td>
</tr>
<tr>
<td>BUFFER</td>
<td>Used for a signal that is an output from an entity. The value of the</td>
</tr>
<tr>
<td></td>
<td>signal can be used inside the entity, which means that in an assignment</td>
</tr>
<tr>
<td></td>
<td>statement, the signal can appear both on the left and right sides of</td>
</tr>
<tr>
<td></td>
<td>the &lt;= operator.</td>
</tr>
</tbody>
</table>

Table A.2. The possible modes for signals that are entity ports.

### VHDL Operators

<table>
<thead>
<tr>
<th>Operator Class</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest precedence</td>
<td>*+, ABS, NOT</td>
</tr>
<tr>
<td>Multiplying</td>
<td>* , /, MOD, REM</td>
</tr>
<tr>
<td>Sign</td>
<td>+, -</td>
</tr>
<tr>
<td>Adding</td>
<td>+, -, &amp;</td>
</tr>
<tr>
<td>Shift</td>
<td>SLL, SRL, SLA, SRA, ROL, ROR</td>
</tr>
<tr>
<td>Relational</td>
<td>=, /=, &lt;, &lt;=, &gt;, &gt;=</td>
</tr>
<tr>
<td>Logical</td>
<td>AND, OR, NAND, NOR, XOR, XNOR</td>
</tr>
</tbody>
</table>

### VHDL Statements

- **Some VHDL constructs:**
  - Signal Assignment: \( A \leftarrow B \);
  - Comparisons:
    - = (equal), > (greater than), < (less than), etc.
  - Boolean operations: AND, OR, NOT, XOR
  - Concurrent statements (when-else)
  - Sequential statements (CASE, IF, FOR)
    - Must be enclosed in a PROCESS block

- **VHDL Reference**
  - Refer to Appendix A in your textbook
  - VHDL Language Guide on class web page

### VHDL Combinational Template

- Every VHDL model is composed of an **entity** and at least one **architecture**

- **Entity** describes the interface to the model (inputs, outputs)

- **Architecture** describes the behavior of the model

- Can have multiple architectures for one entity
A VHDL Template

```
entity model_name is
  port (  
    list of inputs and outputs );
end model_name;
architecture arch_name of model_name is
begin
  concurrent statement 1;
  concurrent statement 2;
  ... concurrent statement N;
end arch_name;
```

- All of the text in blue are VHDL keywords
- VHDL is NOT case sensitive
  – (ENTITY is same as entity is same as EnTiTy)

Example: Majority Function

```
library ieee;
use ieee.std_logic_1164.all;
entity majority is
  port ( A, B, C : in std_logic; -- two dashes is a COMMENT in VHDL
         Y : out std_logic);
end majority;
-- this is the architecture declaration, uses only one concurrent statement.
ARCHITECTURE concurrent of majority is
begin
  Y <= (A and B) or (A and C) or (B and C);
end concurrent;
```

Majority Function with Temporary Signals

```
ARCHITECTURE newconc of majority is
  signal t1, t2, t3 : std_logic;
begin
  t1 <= A and B;
  t2 <= A and C;
  t3 <= B and C;
  Y <= t1 or t2 or t3;
end newconc;
```

Note: temporary signals are declared between architecture statement and begin statement

Majority Function with when-else Statement

```
ARCHITECTURE whenelse of majority is
begin
  Y <= '1' when ( (A and B) or (A and C) or (B and C))
  else '0';
end whenelse;
```

- Many different ways to accomplish the same result in VHDL
- Usually no best way; some may make more sense to you than others
Concurrent vs. Sequential Statements

- The statements we have looked at so far are called concurrent statements
  - Each concurrent statement will synthesize to a block of logic

- Another class of VHDL statements are called sequential statements
  - Sequential statements can ONLY appear inside of a process block
  - A process block is considered to be a single concurrent statement.
  - Can have multiple process blocks in an architecture
  - Usually use process blocks to describe complex combinational or sequential logic

Majority Gate Using `process` Block and `if` Statement

ARCHITECTURE ifstate of majority is
begin
  main: process (A, B, C)
  begin
    Y <= '0'; -- default output assignment.
    if ((A = '1') and (B = '1')) then
      Y <= '1';
      end if;
    if ((A = '1') and (C = '1')) then
      Y <= '1';
      end if;
    if ((B = '1') and (C = '1')) then
      Y <= '1';
      end if;
    end process main;
  end ifstate;
end ifstate;

Comments on `process` Block Model

- The first line in the process "main: process (A, B, C)"
  has the name/label of the process (main) and the sensitivity list of the process
  - The process name is user defined, can also be left out (unnamed process)
  - The sensitivity list should contain any signals that appear on the right hand side of an assignment (inputs) or in any Boolean for a sequential control statement

- The `if` statement condition must return a Boolean value (TRUE or FALSE)
  - the conditional is written as:
    ( (A='1') and (B='1') )
  Cannot write it as:
    (A and B)
  because this will return a 'std_logic' type
Use of if-else

ARCHITECTURE ifelse of majority is
begin
process (A, B, C)
begin
if (((A = '1') and (B = '1')) or ((A = '1') and (C = '1')) or ((B = '1') and (C = '1')) ) then
Y <= '1';
else
Y <= '0';
end if;
end process;
end ifelse;

Comments:
Can have an anonymous process (no name)
Used an 'else' clause to specify what the output should be if the if condition test was not true
CAREFUL! The Boolean operators (OR, AND) do not have any precedence so must use parenthesis to define precedence order

Unassigned Outputs in process Blocks

ARCHITECTURE bad of majority is
begin
process (A, B, C)
begin
if (((A = '1') and (B = '1')) or ((A = '1') and (C = '1')) or ((B = '1') and (C = '1')) ) then
Y <= '1';
else
Y <= '0';
end if;
end process;
end bad;

Comments:
Can have an anonymous process (no name)
Used an 'else' clause to specify what the output should be if the if condition test was not true
CAREFUL! The Boolean operators (OR, AND) do not have any precedence so must use parenthesis to define precedence order

Unassigned Outputs in process Blocks

• A common mistake in writing a combinational process is to leave an output unassigned.

• If there is a path through the process in which an output is NOT assigned a value, then that value is unassigned.

Comments on ‘bad’ Architecture

• In the above process, the ELSE clause was left out. If the 'if' statement condition is false, then the output Y is not assigned a value.
  – In synthesis terms, this means the output Y should have a LATCH placed on it!
  – The synthesized logic will have a latch placed on the Y output; once Y goes to a '1', it can NEVER return to a '0'!!!!!!

More Comments on ‘bad’ Architecture

• This is probably the #1 mistake in writing processes. To avoid this problem do one of the following things:
  – ALL signal outputs of the process should have DEFAULT assignments right at the beginning of the process (probably the easier method).
  – OR, all 'if' statements that affect a signal must have ELSE clauses that assign the signal a value if the 'if' test is false.
Summary

• VHDL is a powerful language that enables multiple methods of describing digital circuits

• Synthesizable VHDL is a subset of VHDL that produces consistent simulation results

• The best way to learn VHDL is by studying examples (textbook, Internet, Quartus II, etc.)