FPGA Design
EECE 277

VHDL Review

Dr. William H. Robinson
April 15, 2005

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

Topics

“Education is a progressive discovery of our own ignorance.”

– Will Durant
(1885-1981)

• Administrative stuff
  – Return graded Exam 2
  – Demo final project on or before April 26 (return UP2 kits)
  – Final project report and presentation due April 30 (exam block)
  – Using the VGA port
  – TA Evaluation

• Exam solutions

• VHDL Review

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

Programming Styles

• Not like C, Fortran, etc.
  – Based on algorithmic sequences of calculations
  – Inherently procedural or serial

• VHDL programs are different
  – Must describe the behavior of a digital circuit
  – Circuit operations will have concurrency
  – Used for both simulation and synthesis
Writing VHDL Code for Synthesis

• **Should not resemble a computer program**
  - Many variables
  - Loops

• **Code easily relates to described logic**
  - If you cannot tell what logic circuit is described, the CAD tool likely won't synthesize the circuit you are trying to model

• **Know your CAD Tool**
  - Understand what certain statements will generate when synthesized

---

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.</td>
</tr>
<tr>
<td>INOUT</td>
<td>Used for a signal that is both an input to an entity and an output from the entity.</td>
</tr>
<tr>
<td>BUFFER</td>
<td>Used for a signal that is an output from an entity.</td>
</tr>
</tbody>
</table>

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

---

VHDL Statements

- **Some VHDL constructs:**
  - Signal Assignment: A <= B;
  - Comparisons
    - = (equal), > (greater than), < (less than), etc.
  - Boolean operations: AND, OR, NOT, XOR
  - Concurrent statements (SELECT, 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
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

Using the **process** Block

• The first line in the process has the name/label of the process 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

Concurrent vs. Sequential Statements

• The order in which you arrange concurrent statements **MAKES NO DIFFERENCE**
  – The synthesized logic will be the same

• Ordering of statements only makes a difference within a **process**
  – This is why statements within a process are called ‘sequential’ statements
  – The logic synthesized reflects the statement ordering (only for assignments to the same output)

Forms of the **Wait** Statement

• **wait for** `<time expression>`
  – Example: wait for gate_delay, wait for 20 ns

• **wait on** `<signal>`
  – Example: wait on R, S, clk

• **wait until** `<boolean expression>`
  – Example: wait until (rising_edge(clk))

• Both sensitivity lists and wait statements cannot be used!
Common Errors in VHDL Code

• Missing semicolon
  – All statements must end with a semicolon;

• Use of quotes
  – Single quotes for single-bit data
  – Double quotes for multi-bit data
  – No quotes for integer data (same for enumerated types)

• Combinational vs. sequential statements
  – Using sequential statements outside of process block

• Implied memory
  – Not assigning all cases or a default case

• Type mismatch
  – VHDL is strongly typed

Multiplexers

• Signals
  – $2^n$ data inputs
  – $n$ control (select) inputs
  – 1 data output

• Binary code on control lines determines which input is connected (gated/routed) to output

• Example: 2:1 Mux

![Graphical symbol](image1)

(a) Graphical symbol
(b) Truth table
(c) Sum-of-products circuit

Decoder

• Takes an $n$-bit number as an input and selects exactly 1 of $2^n$ outputs

• Potential uses:
  – Choosing a memory bank with the decoder as an “enable”
  – Accessing a register location in the register file with the decoder as the “enable”

• Another use for decoders:
  – The output of the decoders are merely the minterms of the inputs
  – You can make any function by merely ORing the appropriate minterms

Demultiplexer

• Routes an input signal to one of $2^n$ outputs

• Implementation is similar to decoder with enable

![Graphical symbol](image2)

(a) Truth table
(b) 1:4 Demultiplexer
Encoders

• Opposite of decoders
  – Encode given information into a more compact form

• Binary encoders
  – \(2^n\) inputs into \(n\)-bit code
  – Exactly one of the input signals should have a value of 1, and outputs present the binary number that identifies which input is equal to 1

• Use: reduce the number of bits (transmitting and storing information)

CAD System

• Design entry
• Synthesis
• Functional simulation
• Physical design
• Timing simulation
• Chip configuration

Code Converter: BCD-to-7-Segment Display

(a) Code converter
(b) 7-segment display
(c) Truth table

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.)