FPGA Design
EECE 277

VHDL Philosophy 101

Dr. William H. Robinson
February 7, 2005

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

Topics

“Good judgment comes from experience. Experience comes from bad judgment.”

– Proverb

• Administrative stuff
  – Turn in Laboratory Assignment #1
  – Feedback from Laboratory Assignment #1
  – Homework Assignment #3 (due Monday, February 14)

• Very High Speed Integrated Circuit (VHSIC)
  Hardware Description Language (HDL)

• VHDL for simulation

• VHDL for synthesis

Describing a System

• System – an assemblage of objects united by some form of regular interaction or interdependence

• What do you think would be important to describe?

Describing a System

• Interface
  – What devices can connect to the system?
  – How is communication handled?
  – What compatibility issues are required?

• Behavior
  – Component chips and interconnections
  – Type of processing on input signals and type of output signals produced
Combinational Logic Networks

- **Functionality**
  - Multiple networks can implement equivalent function

- **Other requirements**
  - Size
  - Power/energy
  - Performance

Mystery Circuit #1

- Sum-of-Products (SOP) CMOS circuit
- Computes the ‘SUM’ function of \( x_1, x_2, x_3 \)

Mystery Circuit #2

- CMOS circuit built with multiplexers
- Computes the ‘SUM’ function of \( x_1, x_2, x_3 \)

Mystery Circuit #3

- CMOS circuit built with XOR gates
- Computes the ‘SUM’ function of \( x_1, x_2, x_3 \)
Implementation in an FPGA

- All three “mystery circuits” would have equal cost in a 3-input LUT

Example: Boolean Functions

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Event-Driven Simulation

- Input transition (0 to 1, 1 to 0) creates an event
- Outputs are evaluated/updated accordingly
- Run simulation until all events have occurred

Propagation Delay

- Real devices require time to switch
- Outputs are initially undefined
- Outputs are updated after finite amount of time
Signal Values

- For digital signals, we expect a binary-valued system of 0 or 1
  - Corresponds to signals driven to V_{DD} or GND

- What happens if a signal is disconnected?
  - High-impedance

- What happens if a signal is driven to 0 or 1 simultaneously?
  - Design error

- What happens when the initial value is undefined?
  - Propagation delays for circuit start-up

IEEE 1164 Standard Signal Definitions

- **U**: Uninitialized
- **X**: Forcing Unknown
- **0**: Forcing 0
- **1**: Forcing 1
- **Z**: High impedance
- **W**: Weak Unknown
- **L**: Weak 0
- **H**: Weak 1
- **-**: Don’t Care
Hardware Description Languages

- **Structural description**
  - A connection of components

- **Functional description**
  - A set of Boolean formulas, state transitions, etc.

- **Simulation description**
  - A program designed for simulation

- **Major languages**
  - Verilog
  - VHDL

Verilog vs. VHDL

- **System**
  - Relatively easy to learn
  - Fixed data types
  - Interpreted constructs
  - Good gate-level timing
  - Limited design reusability
  - No structure replication

- **Behavioral (Algorithms)**
  - Relatively difficult to learn
  - Abstract data types
  - Compiled constructs
  - Less good gate-level timing
  - Good design reusability
  - Supports structure replication

- **Functional (RTL, Boolean)**

- **Structural (Gate, Switch)**

Abstraction Levels in Digital Systems

- **Textual languages for describing hardware**
  - Structure
  - Function

- **Most people today use textual languages rather than schematics for most digital design**
  - Schematics (transistor or gate-level) make poor use of screen space
  - Still may use block diagrams for higher levels
Advantages of Using HDLs

• **Design portability**
  – Circuits can be implemented in different chips (technology independence)
  – Code can be used by different CAD tools (vendor independence)

• **Design sharing and reuse**
  – Allows documentation of source code
  – Faster development of new products (prototyping)

• **Modularity**
  – Allows hierarchical design
  – Multiple instances of components can be used

---

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

---

Synthesis Subsets

• **VHDL and Verilog were designed for simulation**

• **A synthesis subset is**
  – Synthesizable
  – Produces consistent simulation results

• **Different tools may use different synthesis subsets**

• **Textbook by Yalamanchili**

---

Simulation vs. Synthesis

• **Complementary processes**
  – In either case, start with the specification of the digital system
  – Construct a VHDL model

• **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
Functional Simulation

Testbenches

- A testbench is a model used to exercise a simulation
  - Provides stimulus
  - Checks outputs

- Testbenches help automate design verification
  - Rerun edited module against testbench
  - Run models (behavioral, RTL) against the same testbench

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

Sample VHDL

2.46. The function can be specified by using the minterms as follows:

```vhdl
ENTITY problem46 IS
PORT (x1, x2, x3 : IN BIT ;
f : OUT BIT );
END problem46 ;
ARCHITECTURE LogicFunc OF problem46 IS
BEGIN
f <= (NOT x1 AND NOT x2 AND NOT x3) OR (NOT x1 AND NOT x2 AND x3) OR
(x1 AND NOT x2 AND NOT x3) OR (x1 AND NOT x2 AND x3) OR
(x1 AND x2 AND NOT x3) OR (x1 AND x2 AND x3) ;
END LogicFunc ;
```

- What are some keywords in VHDL?
VHDL Design Entity

- **Interface**
  - Connections to the system

- **Behavior**
  - Type of processing on input signals and type of output signals produced

### 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>Multiply</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, NXOR</td>
</tr>
</tbody>
</table>

### VHDL Code for a Full-Adder

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY fulladd IS
    PORT ( Cin, x, y : IN STD_LOGIC;
            s, Cout : OUT STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
    s <= x XOR y XOR Cin ;
    Cout <= (x AND y) OR (x AND Cin) OR (y AND Cin) ;
END LogicFunc ;
```
Hardware Inference for Synthesis

- **Operators**
  - Must be inferred to process signals

- **Storage**
  - Must be inferred when a signal may retain its previous value

- **Building blocks**
  - Synthesis tool must compile design to the available components (standard cells, LEs, etc.)

Summary

- All functions that fit in an LE have the same cost

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

- Testbenches can be used to exercise design models