Digital Design with CAD Tools

**Subsystem Function & Interface Specification**

- Concept Design
- Design Entry
  - Behavioral Description
  - Logic Synthesis
    - RTL Logic Gate Networks
    - Functional (Behavioral) Simulation
      - Verify functions as specified
      - May use idealized models
    - Timing Diagrams
  - Test Vectors
- IMPLEMENTATION

**Design Entry**

Two Major Methods:
- Schematic Capture
  - Graphical Description
- Hardware Description Language
  - Verbal Description of Function
  - Provides many of the same constructs as other programming languages (if-then-else, case, for loops, while, …)

**Hardware Description Languages**

Two Standard (IEEE) Types:
- Verilog HDL
- VHDL
  - Very high-speed integrated circuit Hardware Description Language

Digital Design with CAD Tools

**IMPLEMENTATION**

- Design Mapping (Targeting)
  - Pinouts
  - Physical Layout / Signal Routing
  - Test Vectors
    - Timing Simulation
    - Physical Models
  - Fabricate or Program Device
  - Test Hardware

- IMPLEMENTATION
  - Fit and adapt the design for implementation in the available circuits in the target device
  - Define which circuits are used or where they are placed.
  - Define interconnect routes between devices
  - Verify still functions as specified
  - Using true circuit/signal delay models, capacitances, etc.
Hardware Description Languages

Advantages:
• Portability
  – Designs described in “standard” form used by many vendors’ tools

• Technology Independence
  – Descriptions are not tied to any particular physical implementation technology or methodology
  – Same code can be used for variety of implementations (PLDs, ASICs, ...)

• Design Reuse
  – Supports a modular approach to designing systems
  – Finished, working, proven modules can be used again in later designs

Writing VHDL Code

General Guidelines:
• Make your VHDL code “self documenting”
  – Use names for signals and devices that convey what they are or what they do
  
  Example: enable_output vs. X1
  - VHDL is __________________ (En_OUT = en_out)
  - Identifiers can have letters, numbers or an underscore “_” (but not 2 consecutive underscores)
  - Identifiers MUST start with a ____________, and can NOT end with an underscore

Examples

<table>
<thead>
<tr>
<th>Valid Identifiers</th>
<th>Invalid Identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_bus_val</td>
<td>dBus_val</td>
</tr>
<tr>
<td>descriptive name</td>
<td>$Bus_val</td>
</tr>
<tr>
<td>begins with numeric character</td>
<td></td>
</tr>
<tr>
<td>WE</td>
<td>DDD ddd</td>
</tr>
<tr>
<td>classic “write enable” acronym</td>
<td></td>
</tr>
<tr>
<td>div_flag</td>
<td>mid_num</td>
</tr>
<tr>
<td>a real winner</td>
<td>contains illegal character</td>
</tr>
<tr>
<td>port_A</td>
<td>last_value</td>
</tr>
<tr>
<td>provides some info</td>
<td>contains consecutive underscores</td>
</tr>
<tr>
<td>in_bus</td>
<td>start_val</td>
</tr>
<tr>
<td>input bus (a good guess)</td>
<td>ends with underscore</td>
</tr>
<tr>
<td>clk</td>
<td>in</td>
</tr>
<tr>
<td>classic system clock name</td>
<td>uses VHDL reserved word</td>
</tr>
<tr>
<td>Big_VALUE</td>
<td>$####$</td>
</tr>
<tr>
<td>valid way too ugly</td>
<td></td>
</tr>
<tr>
<td>pa</td>
<td>sim-val</td>
</tr>
<tr>
<td>possibly lacks meaning</td>
<td>illegal character (dash)</td>
</tr>
</tbody>
</table>

Valid and Invalid VHDL Identifiers

Ref: Low-Carb VHDL Tutorial – B. Mealy
Writing VHDL Code

General Guidelines:

• Make your VHDL code “___________________”
  – Use names for signals and devices that convey what they are or what they do
    Example: enable_output vs. x1

• Use comments to clarify for someone else reading your code
  – Comments begin with: - - (2 dashes)

• VHDL ignores “white space” (spaces, tabs,..)
  – Use indenting to help make code readable

VHDL Basics

• ______________________
  – “Black Box” description of circuit that declares inputs and outputs, their type, and their size
  – Interface specification

• ______________________
  – what’s inside the box
  – Specifies the implementation of your circuit

VHDL Entity

ENTITY modulename IS
  PORT (  input1 : IN signaltype;
          input2 : IN signaltype;
          output1 : OUT bus_type;
          output2 : OUT signaltype );
END modulename ;

Signal Types:

Single signal: ____________________ ( Has values: 0, 1, Z, - (don’t care), etc.)

Multiple (8) signal bus: ____________________

If we use the signal type “STD_LOGIC” or “STD_LOGIC_VECTOR”, we must include 2 statements at the top of our VHDL code:

LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY ieee
USE ieee.std_logic_1164.all

ENTITY Hazardous_Ckt

PORT: IN
A
C
B

PORT: OUT
F

END ENTITY;

ARCHITECTURE arch_name OF entity_name IS

-- Example of an internal signal declaration:
SIGNAL int_sig1: STD_LOGIC;

SIGNAL int_sig2: STD_LOGIC;

BEGIN
concurrent statement1;
concurrent statement2;
concurrent statement3;
END arch_name;

Architecture Declaration:
• name the architecture
• associate it with a particular "Entity"
• describe any important internal signals

Architecture Body:
Where we describe the internal workings

Internal Signal Declarations

ARCHITECTURE arch_name OF entity_name IS

-- Example of an internal signal declaration:
SIGNAL int_sig1: STD_LOGIC;

SIGNAL int_sig2: STD_LOGIC;

BEGIN
concurrent statement1;
concurrent statement2;
concurrent statement3;
END arch_name;

Concurrent Statements:

• These all execute ________________.

  – Treated like they are all ________________

• The order that these statements appear in the VHDL code (usually) ________________.

  – So don’t count on one being done before executing the next.
Simple Concurrent Signal Assignments

Signal Assignment Operators:

- "C" Language:     =
- VHDL:        <=

Simple Concurrent Signal Assignment:

```
target <= expression
```

--Can do Boolean logic

Concurrent Signal Assignments

ENTITY entity_name IS
  PORT ( input1, input2 : IN STD_LOGIC;
         output2 : OUT STD_LOGIC;
         input3_bus : IN STD_LOGIC_VECTOR( 3 DOWNTO 0 ) ; -- 4 bits
         output1_bus : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0) ); -- 8 bits
END entity_name;

ARCHITECTURE arch_name OF entity_name IS
  SIGNAL internal_sig : STD_LOGIC;
BEGIN
  internal_sig <= input1 AND input2 ; --Can do Boolean logic
  output2  <= _______ ;  --Assigns a single bit binary value to a signal
  output1_bus <= __________ ;  --Assigns multi-bit binary values to a vector
  output3 <= input2 XOR input3_bus_____ ;  --References 1 bit of a vector
END arch_name;

Simple Concurrent Signal Assignments

Boolean Logic Operators Available:

AND, OR, NOT, NAND, NOR, XOR, XNOR

Precedence:

- VHDL does **NOT** assume any logical operator precedence (except “NOT” above others)
- You must use parentheses to clarify for VHDL

```
F <= A AND B OR NOT C AND D ; --?????
F <= (A AND B) OR (NOT C AND D) ; -- OK
```