

## ECE 551: Digital System Design & Synthesis

VHDL versus Verilog  
Lectures Set 8 – Part 1  
VHDL  
(1 Lecture)

## Overview

- VHDL – Language introduction
- History of VHDL and Verilog
- Comparison using different metric (Ref: CRK)
- Comparison Summary
- Algorithm and RTL Level Examples (Smith)

3/19/2002

2

## VHDL – Language introduction

- An example
- Abstraction Levels
- VHDL Constructs
- Entities and Architectures
- Signals and Variables
- Concurrent VHDL
- Sequential VHDL
- Structural VHDL
- VHDL Types
- Libraries and Packages
- A Bit on Clocking

3/19/2002

3

## An Example

```
entity FullAdder is -- Declaration of entity named FullAdder
  port (X, Y, Cin: in bit; -- 1-bit inputs X, Y and Cin
        Cout, Sum: out bit; -- Two 1-bit outputs
  end decoder_3_to_8; -- End of entity declaration

architecture equation of FullAdder is -- Architecture named functional
  begin -- for entity FullAdder
    Sum <= X xor Y xor Cin -- equation for Sum
    Cout <= (X and Y) or (X and Cin) or (Y and Cin) -- equation for Cout
  end equation;
```

3/19/2002

4

## An Example (Contd.)

```
entity ParallelAdder is -- Declaration of entity named FullAdder
  port (A, B: in bit_vector(3 downto 0); -- 4-bit inputs A, B
        Ci: in bit; -- one bit input Cin
        S: out bit_vector(3 downto 0); -- 4 bit output S
        Co: out bit; -- 1-bit output Cout
  end ParallelAdder; -- End of entity declaration

architecture structure of ParallelAdder is -- Architecture named structure
  component FullAdder -- can use any component from library
    port (X, Y, Cin: in bit;
          Cout, Sum: out bit);
  end component
```

3/19/2002

5

## An Example (Contd.)

```
signal C; bit_vector(3 downto 1);

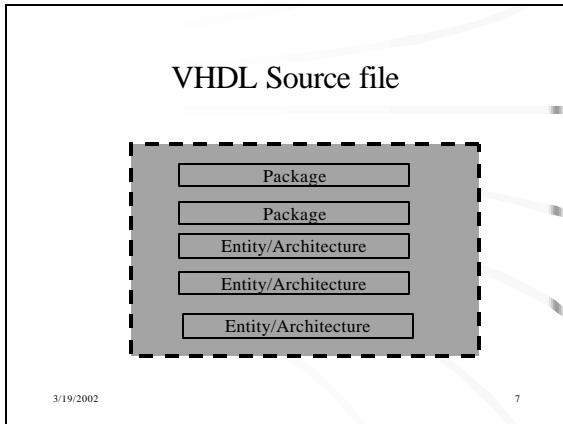
begin -- instantiate FullAdder

  FA0: FullAdder port map (A(0), B(0), Ci, C(1), S(0)); -- instantiation
  FA1: FullAdder port map (A(1), B(1), C(1), C(2), S(1)); -- instantiation
  FA2: FullAdder port map (A(2), B(2), C(2), C(3), S(2)); -- instantiation
  FA3: FullAdder port map (A(3), B(3), C(3), Co, S(3)); -- instantiation

end structure;
```

3/19/2002

6



- ### VHDL Abstraction Levels
- **Functional:**
    - Algorithms without timing
  - **Behavioral:**
    - Algorithms (Behavior) with timing. Register level implementation not defined
  - **RTL (Register Transfer Level):**
    - Register level implementation defined in terms of registers, datapaths, state machines
  - **Logic or Gate Level:**
    - Implementation defined in terms of logic components such as gate s, flip-flops, and Boolean equations – example shown earlier
    - There are no primitive gates as in Verilog – only gate functions exist
- 3/19/2002 8

- ### VHDL Constructs
- **Concurrent VHDL or Dataflow:**
    - Consists of a collection of statements and processes that execute concurrently
  - **Sequential VHDL:**
    - Consists of the sequences of statements within processes
    - Logic described may be combinational or sequential
  - **Structural:**
    - Describes interconnections of components (entities and components) –components at lowest level may be defined elsewhere
    - Analogous to logic diagrams or netlists
- 3/19/2002 9

- ### Entities and Architectures
- **Entity:**
    - The primary hardware abstraction in VHDL
    - Represents a part of a hardware design that has well-defined inputs and outputs and a well-defined function
    - Provides: the entity name, the inputs and outputs
    - Analogous to a symbol in a block diagram
  - **Architecture Body:**
    - Specifies the relationships between the inputs and outputs of a design entity
    - May be a mixture of structural, concurrent and sequential VHDL.
  - A given entity may have multiple different architectures.
- 3/19/2002 10

### Entities and Architectures - An Example

```

library ieee;           -- Instantiations of a library and a package are tied
use ieee.std_logic_1164.all; -- to the following entity only.
entity decoder_3_to_8 is -- Declaration of entity named decoder_3_to_8
  port (A : in std_logic_vector(2 downto 0); -- 3-bit input A of type std_logic
        D : out std_logic_vector(7 downto 0)); -- 8-bit output D of type std_logic
end decoder_3_to_8;     -- End of entity declaration

architecture functional of decoder_3_to_8 is -- Architecture named functional
  begin -- for entity decoder_3_to_8
    D <= B"00000001" when A = B"000" -- Concurrent when else statement
        else B"00000010" when A = B"001" -- for evaluation of output D.
        else B"00000100" when A = B"010"
        else B"00001000" when A = B"011"
  end
  
```

3/19/2002 11

### Entities and Architectures – An Example (Continued)

```

    else B"00010000" when A = B"100"
    else B"00100000" when A = B"101"
    else B"01000000" when A = B"110"
    else B"10000000" when A = B"111"
    else B"XXXXXXXX" when others; -- For combinations involving std_logic
end functional; --values in addition to 0 and 1.
  
```

3/19/2002 12

## Signals and Variables

- A **signal** can be viewed as a wire or cluster of wires
- Signals are concurrent and sequential objects
- A **variable** is not viewed as wire
- A variable is only a sequential object

3/19/2002

13

## Signals and Variables – Declaration

- VHDL is case-insensitive: OR, Or and or are equivalent
- VHDL keywords are highlighted using boldface type
  - Keywords may not be used for identifiers, etc.
- Port declarations for entities are signal declarations
- Signal declarations use modes and typing
- Variable declarations use typing
- Variables declared within processes, functions, or procedures

3/19/2002

14

## Signals and Variables– Assignment

- Signal assignment:
  - $F \leq A \text{ and } B;$
- Variable assignment:
  - $F := A \text{ and } B;$
- Variables can be used only in a process (sequential VHDL).
- Variables are immediately evaluated with new value available to the next statement.
- Process has sensitivity list.

3/19/2002

15

## Concurrent VHDL

- $\leq$   
 $z \leq a \text{ or } b;$
- **when else**  
 $F \leq X \text{ or } Y \text{ when } S = "00"$   
 $\text{ else } X \text{ xor } Y \text{ when } S = "01"$   
 $\text{ else } X \text{ and } Y \text{ when } S = "10"$   
 $\text{ else } Y \text{ when } T = '1';$  – Statements are evaluated in order  
 $\text{ else } X;$
- **with select**  
**with S select** – Limited to single selection expression  
 $F \leq X \text{ or } Y \text{ when } "00"$  – So less flexible than when else  
 $X \text{ xor } Y \text{ when } "01"$   
 $X \text{ and } Y \text{ when } "10"$   
 $\text{ not } X \text{ when } "11"$   
 $"XXXX" \text{ when others};$

3/19/2002

16

## Sequential VHDL

- Used within processes, functions and procedures
- Statements are executed sequentially as in the typical programming language
- Processes execute concurrently with each other
- A complete process behaves like a concurrent statement
- In a process, **variables** can be used as well as signals
- Variables are evaluated without delay, i.e., immediately
- The result of a variable evaluation is available to subsequent statements in the process
- Variables are declared and assigned only within sequential VHDL
- Variables may be equated to signals and signals to variables as long as they are of the same type

3/19/2002

17

## Sequential VHDL – The Process

- A sequence of sequentially -executed statements.
  - The statements are ordered
- A **sensitivity list** governs the activation of the process.
  - The process executes only when a signal on the sensitivity list changes
  - If process represents a combinational circuit, all inputs must be on the sensitivity list
  - If process represents the state of a synchronous state machine, the clock and asynchronous reset signal typically are the sensitivity list

3/19/2002

18

## Sequential VHDL - A Process Example

```

pick: process (S,D) is -- Process statement with sensitivity list
begin
  case S is -- Case statement conditioned by S
    when B"00" => Y <= D(0); -- Use of subarrays D(i) of
    when B"01" => Y <= D(1); -- array D.
    when B"10" => Y <= D(2);
    when B"11" => Y <= D(3);
    when others => Y <= 'X';
  end case;
end process pick;

```

3/19/2002

19

## Sequential VHDL

### – if then else

```

if ...
then ...
elsif ...
then ...
...
else
endif

```

3/19/2002

20

## Sequential VHDL - case

- Synthesizes more like a regular “flat decoder.”
- Example – Case:

```

case S is -- Case statement conditioned by S
  when B"00" => Y <= D(0);
  when B"01" => Y <= D(1);
  when B"10" => Y <= D(2);
  when B"11" => Y <= D(3);
  when others => Y <= 'X';
end case;

```

3/19/2002

21

## Sequential VHDL – Other Constructs

- Loops
  - for
  - while
- Statements
  - wait
  - null
    - Used on right hand side to leave signals unchanged
    - Example: acc <= null when others;
- Subprograms
  - function
  - procedure

3/19/2002

22

## Structural VHDL

- Structural VHDL is based on an interconnection of components (entities).
- A key element of structural VHDL is the **component**.
  - Within an entity, declaration of another entity to be instantiated
  - Declaration of component must match referenced entity
  - In VHDL 87, use of components is required
  - In VHDL 93, the component not required; can use direct instantiation
- No built-in logic gates

3/19/2002

23

## Structural VHDL – Components (contd.)

- Example: Component Declaration
 

```

component full_adder
  port (X, Y, Z : in std_logic;
        S, C : out std_logic);
end component;

```
- Example: Component Instantiation
  - Explicit
 

```

fa0: full_adder
  port map (X => A(0), Y => B(0), Z => C0, S => S(0), C => C(1));

```
  - Implicit
 

```

fa0: full_adder
  port map (A(0), B(0), C0, S(0), C(1)); -- order dependent

```

3/19/2002

24

## VHDL Types

- VHDL Operators highly restricted to types
  - +, -, : integer; or, and : bit, bit\_vector
- IEEE 1076 Standard Package Types:
  - type bit is ('0', '1');
  - type bit\_vector is array (integer range <>) of bit;
  - type integer is range MININT to MAXINT;
  - subtype positive is integer range 1 to MAXINT;
  - subtype natural is integer range 0 to MAXINT;
  - type boolean is (TRUE, FALSE);

3/19/2002

25

## VHDL Types (Continued)

- IEEE 1164 Package – allows simulation values beyond 0 and 1
- IEEE 1164 Types: (allows 7 added values for simulation)
  - type std\_ulogic is ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-');
  - type std\_ulogic\_vector is array (natural range <>) of std\_ulogic ;
  - subtype std\_logic is resolution\_func std\_ulogic;
  - type std\_logic\_vector is (natural range <>) of std\_logic;
  - subtype X01Z is resolution\_func std\_logic range 'X' to 'Z';
  - -- includes X, 0, 1, Z
- std\_logic is widely used

3/19/2002

26

## Libraries

- Compiled VHDL entities stored in a **library** with a logical name.
  - These entities can be components or packages.
- Library or libraries containing components or packages to be used must be specified for each of the entities in a description.
- By default, std and work libraries are specified for each entity declared
- Typically specify IEEE library and user libraries (if any):
 

```
library ieee;
library lcdf; -- user library for Mano & Kime textbook
```

3/19/2002

27

## Packages

- A package contains functions, procedures, types, constants, attributes or components
- A package consists of a declaration and a body
- Declaration is similar to multiple entities, but describes instead exportable objects
- Body is similar to multiple architectures, providing the actual subprograms and component bodies
- By default, the following package use is specified:
  - **use std.standard.all;**
- Typically specify IEEE packages and user packages (if any):
 

```
use ieee.std_logic_1164.all;
use lcdf.gates; -- gate package in user library for Mano & Kime
```

3/19/2002

28

## Packages (continued)

- Other packages may be essential to the designs you do to avoid difficult VHDL coding, e. g., typing problems.
- To handle +, - and **and**, **or** for the same signals, use IEEE packages:
 

```
ieee.std_logic_arith
ieee.std_numeric
```

3/19/2002

29

## A Bit on Clocking

- Clocking uses notion of **event**, i. e., a change in signal value.
- Example – Positive Edge-Triggered Flip-Flop:
 

```
signal D, clk, Q: std_logic;
process (clk, reset) is
begin
  if (reset = '1')
    Q <= '0';
  elsif (clk'event and clk = '1') then
    Q <= D;
  end if;
end process;
```

3/19/2002

30

## References

- *IEEE Standard VHDL Language Reference Manual*, ANSI/IEEE Std 1076-1993, Institute of Electrical and Electronic Engineers, Inc., 1994.
- *HDL Synthesis Guide* (Version 4.2), Exemplar Logic, Inc., 1998.
- Ashenden, P., *The Designer's Guide to VHDL*, Morgan Kaufmann, 1996 (paperback).
- Sjöholm, S., and L. Lindh, *VHDL for Designers*, Prentice Hall Europe, 1997.
- Mano, M., and C. Kime, *Logic and Computer Design Fundamentals*, 2nd Edition, Prentice-Hall, 2000.
- C. H. Roth, *Digital System Design using VHDL*, PWS Publishing Company, 1998