Introduction to VHDL

Wenye Li, Ph.D
The Chinese University of Hong Kong, Shenzhen

Note

- A short discussion of VHDL within <3 hours.
  - A full investigation needs one semester or more.
- Two ways learning a programming language:
  - Learn through grammars (slow)
  - Learn through examples/practices (quick)
  - This course, firstly learn by examples, then grammars
- Some materials are NOT directly from the textbook!

Outline

- **VHDL Examples**
- VHDL Language
- Advanced Examples

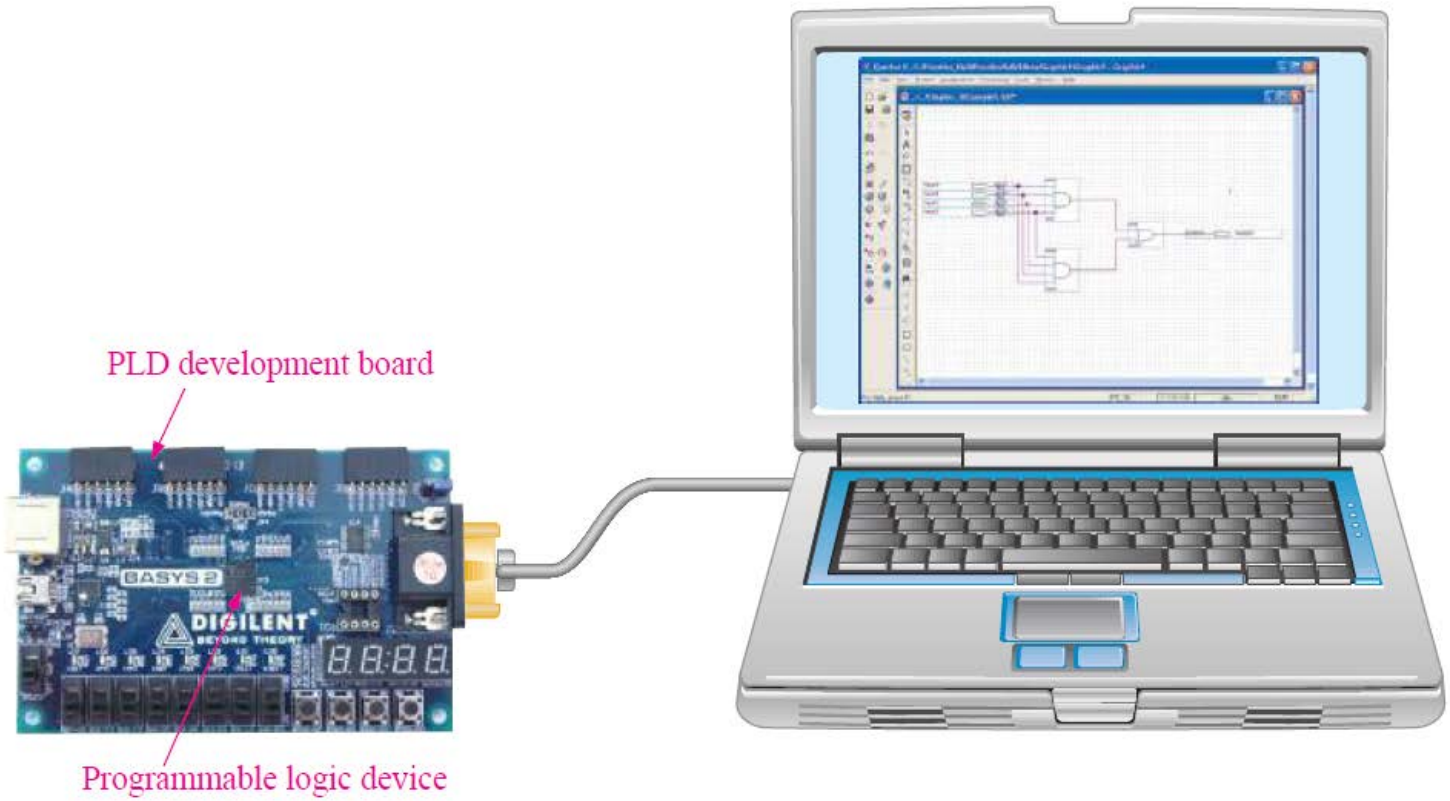


FIGURE 3–55 Programming setup for reprogrammable logic devices. (Photo courtesy of Digilent, Inc.)

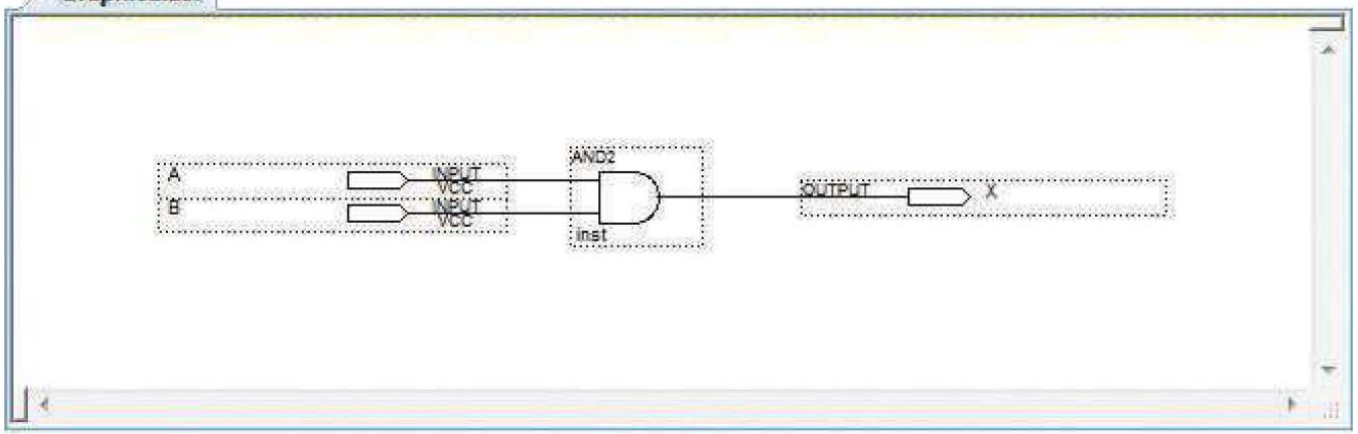
Text Entry and Graphic Entry

Vhdl1.vhd

```
entity VHDL1 is
   port(A, B: in bit; X: out bit);
end entity VHDL1;
architecture ANDfunction of VHDL1 is
begin
   X <= A and B;
end architecture ANDfunction;
```

(a) VHDL text entry

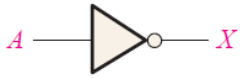
Graphic1.bsf



(b) Equivalent graphic (schematic) entry

FIGURE 3–56 Examples of design entry of an AND gate.

Basic Gates



$X = \overline{A}$

```
entity Inverter is
   port (A: in bit; X: out bit);
end entity Inverter;
architecture NOTfunction of Inverter is
begin
   X <= not A;
end architecture NOTfunction;
```

(a) Inverter

$X = AB$

```
entity ANDgate is
   port (A, B: in bit; X: out bit);
end entity ANDgate;
architecture ANDfunction of ANDgate is
begin
   X <= A and B;
end architecture ANDfunction;
```

(b) AND gate

$X = A + B$

```
entity ORgate is
   port (A, B: in bit; X: out bit);
end entity ORgate;
architecture ORfunction of ORgate is
begin
   X <= A or B;
end architecture ORfunction;
```

(c) OR gate

$X = \overline{ABC}$

```
entity NANDgate is
   port (A, B, C: in bit; X: out bit);
end entity NANDgate;
architecture NANDfunction of NANDgate is
begin
   X <= A nand B nand C;
end architecture NANDfunction;
```

(d) NAND gate

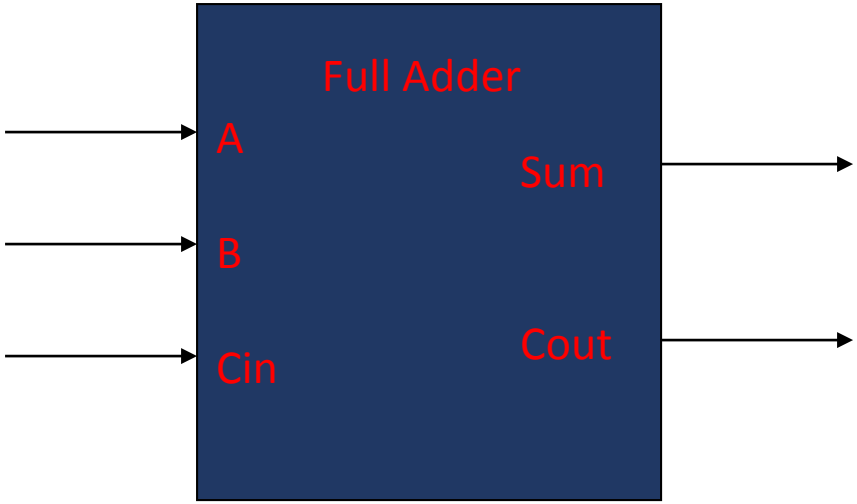
$X = \overline{A\overline{B} + \overline{A}B}$

```
entity XNORgate is
   port (A, B: in bit; X: out bit);
end entity XNORgate;
architecture XNORfunction of XNORgate is
begin
   X <= A xnor B;
end architecture XNORfunction;
```

(e) XNOR gate

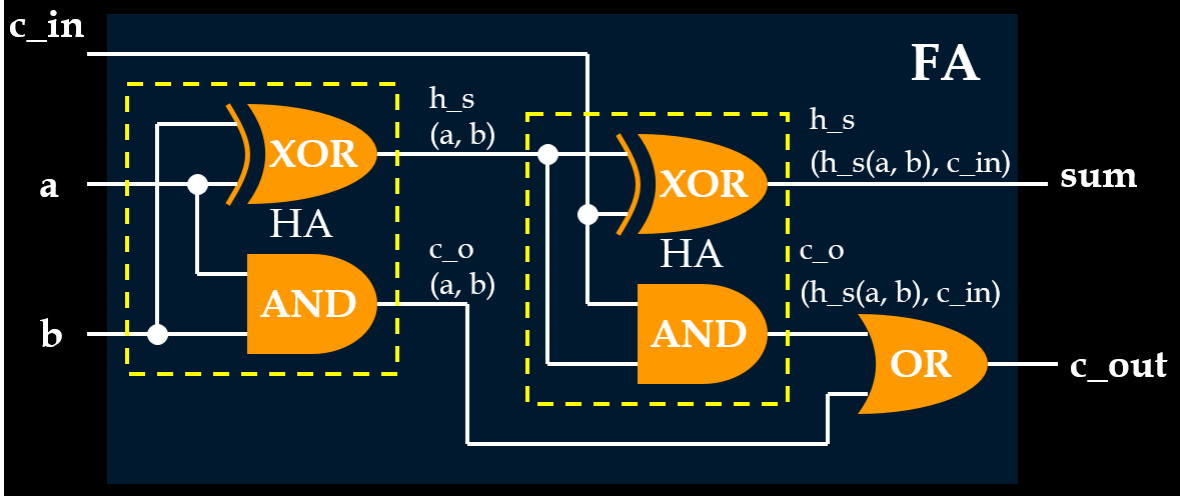
FIGURE 3–58 Logic gates described with VHDL.

Entity



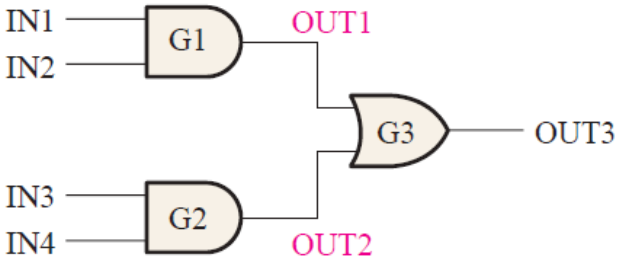

```
ENTITY Full_adder IS
   PORT (                          -- I/O ports
            a:     IN STD_LOGIC;       -- a input
            b:     IN STD_LOGIC;       -- b input
            cin:   IN STD_LOGIC;       -- carry input
            sum:   OUT STD_LOGIC;   -- sum output
            cout: OUT STD_LOGIC);   -- carry output
END Full_adder ;
```

Architecture




```
ARCHITECTURE dataflow OF Full_adder IS
BEGIN
  sum <= a xor b xor cin;
  cout <= (a and b) or (a and cin) or
                    (b and cin);
END dataflow;
```

Component




```
entity AND_OR_Logic is
   port (IN1, IN2, IN3, IN4: in bit; OUT3: out bit);
end entity AND_OR_Logic;

architecture LogicOperation of AND_OR_Logic is

   component AND_gate is
      port (A, B: in bit; X: out bit);
   end component AND_gate;

   component OR_gate is
      port (A, B: in bit; X: out bit);
   end component OR_gate;

   signal OUT1, OUT2: bit;

begin
   G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
   G2: AND_gate port map (A => IN3, B => IN4, X => OUT2);
   G3: OR_gate port map (A => OUT1, B => OUT2, X => OUT3);
end architecture LogicOperation;
```

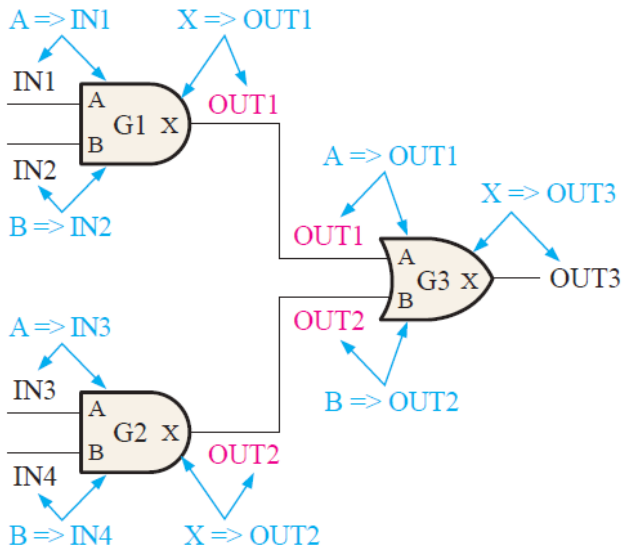
Component declaration for the AND gate

Component declaration for the OR gate

Signal declaration

Component instantiations describe how the three gates are connected.

Component Initialization



G1: AND_gate **port map** (A => IN1, B => IN2, X => OUT1);

G2: AND_gate **port map** (A => IN3, B => IN4, X => OUT2);

G3: OR_gate **port map** (A => OUT1, B => OUT2, X => OUT3);

Component instantiations describe how the three gates are connected.

Exercise

Write a VHDL program for the SOP logic circuit in Figure 5–41 using the structural approach and compare with the data flow approach. Assume that VHDL components for a 3-input NAND gate and for a 2-input NAND are available. Notice the NAND gate G4 is shown as a negative-OR.

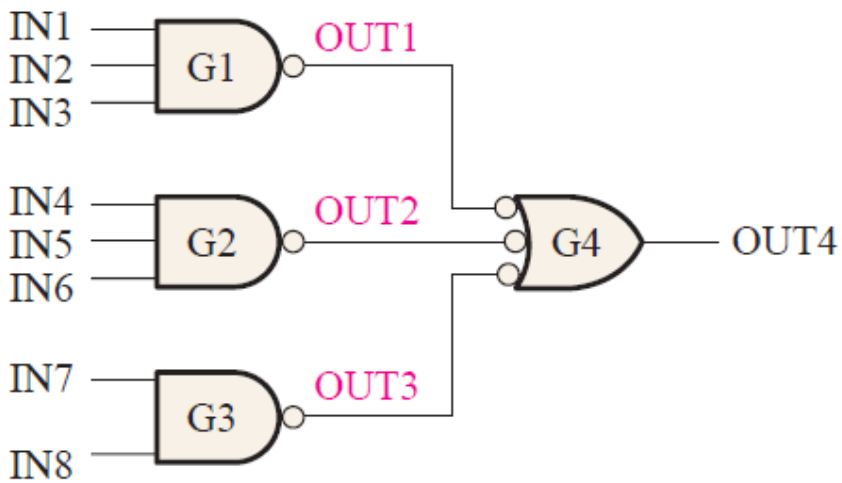


FIGURE 5–41

Structural Approach

```
entity SOP_Logic is
   port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit; OUT4: out bit);
end entity SOP_Logic;
architecture LogicOperation of SOP_Logic is
--component declaration for 3-input NAND gate
   component NAND_gate3 is
      port (A, B, C: in bit X: out bit);
   end component NAND_gate3;
--component declaration for 2-input NAND gate
   component NAND_gate2 is
      port (A, B: in bit; X: out bit);
   end component NAND_gate2;
   signal OUT1, OUT2, OUT3: bit;

begin
G1: NAND_gate3 port map   (A => IN1, B => IN2, C => IN3, X => OUT1);
G2: NAND_gate3 port map   (A => IN4, B => IN5, C => IN6, X => OUT2);
G3: NAND_gate2 port map   (A => IN7, B => IN8, X => OUT3);
G4: NAND_gate3 port map   (A => OUT1, B => OUT2, C => OUT3, X => OUT4);

end architecture LogicOperation;
```

Data Flow Approach

```
entity SOP_Logic is
   port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit; OUT4: out bit);
end entity SOP_Logic;

architecture LogicOperation of SOP_Logic is
begin

   OUT4 <= (IN1 and IN2 and IN3) or (IN4 and IN5 and IN6) or (IN7 and IN8);
end architecture LogicOperation;
```

VHDL Development Environment

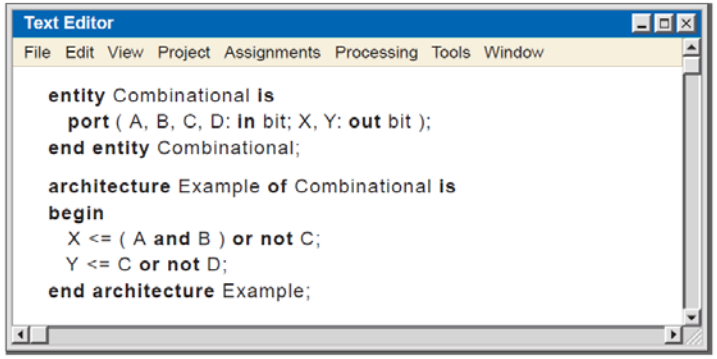


FIGURE 5–42 A VHDL program for a combinational logic circuit after entry on a generic text editor screen that is part of a software development tool.

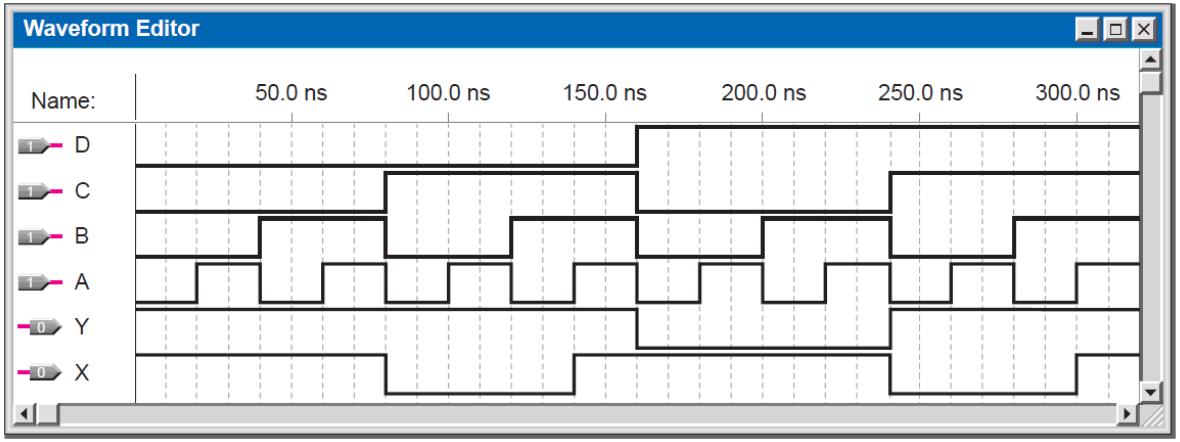


FIGURE 5–43 A typical waveform editor tool showing the simulated waveforms for the logic circuit described by the VHDL code in Figure 5–42.

Outline

- VHDL Examples
- **VHDL Language**
- Advanced Examples

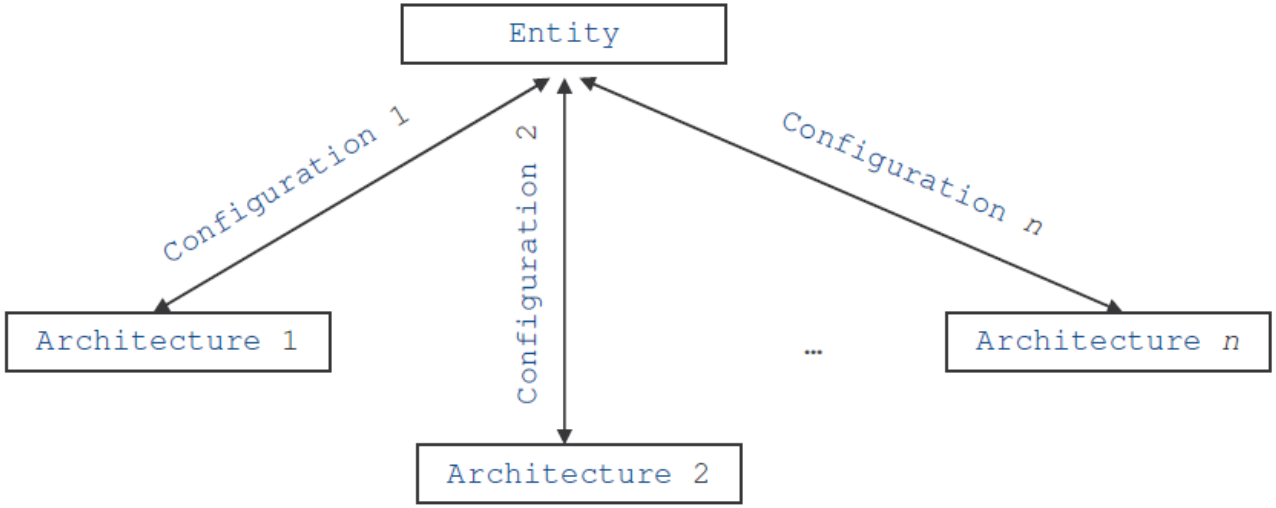
Libraries and Packages

- Libraries provide a set of packages, components, and functions that simplify the task of designing hardware
- Packages provide a collection of related data types and subprograms
- The following is an example of the use of the `ieee` library and its `std_logic_1164` package:

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;
```

Entities, Architectures and Configurations

- The structure of a VHDL design resembles the structure of a modern, object-oriented software design
- All VHDL designs provide an external interface and an internal implementation
- A VHDL design consists of entities, architectures, and configurations



Entities

- An entity is a specification of the design's external interface
- Entity declarations specify the following:
  1. The name of the entity
  2. A set of generic declarations specifying instance-specific parameters
  3. A set of port declarations defining the inputs and outputs of the hardware design
- Generic declarations and port declarations are optional

Entities

- Entity declarations are specified as follows:

```
ENTITY entity_name IS
    GENERIC(
            generic_1_name      : generic_1_type;
            generic_2_name      : generic_2_type;
            generic_n_name      : generic_n_type
    );
    PORT(
            port_1_name   : port_1_dir  port_1_type;
            port_2_name   : port_2_dir  port_2_type;
            port_n_name   : port_n_dir  port_n_type
    );
END entity_name;
```

Ports

- Port name choices:
  - Consist of letters, digits, and/or underscores
  - Always begin with a letter
  - Case insensitive
- Port direction choices:

`IN`	Input port
`OUT`	Output port
`INOUT`	Bidirectional port
`BUFFER`	Buffered output port

NOTE:

A buffer is an output that can be "read" by the architecture of the entity.

Ports

- IEEE standard 1164-1993 defines a package which provides a set of data types that are useful for logic synthesis
  - The external pins of a synthesizable design must use data types specified in the `std_logic_1164` package
  - IEEE recommends the use of the following data types to represent signals in a synthesizable system:

    `std_logic`

    `std_logic_vector(<max> DOWNTO <min>)`

Architecture

- An architecture is a specification of the design's internal implementation
- Multiple architectures can be created for a particular entity
- For example, you might wish to create several architectures for a particular entity with each architecture optimized with respect to a design goal:
  - Performance
  - Area
  - Power Consumption
  - Ease of Simulation

Architecture

- Architecture declarations are specified as follows:

```
ARCHITECTURE architecture_name OF entity_name IS
BEGIN
    -- Insert VHDL statements to assign outputs to
    -- each of the output signals defined in the
    -- entity declaration.
END architecture_name;
```

Configurations

- A configuration is a specification of the mapping between an architecture and a particular instance of an entity
- By default, a configuration exists for each entity
- The default configuration maps the most recently compiled architecture to the entity
- Configurations are most often used to specify alternative architectures for hardware designs

Signals

- Signals represent wires and storage elements
- Signals may only be defined inside architectures
- Signals are associated with a data type
- Signals have attributes
- VHDL is a strongly-typed language:
  - Explicit type conversion is supported
  - Implicit type conversion is not supported

Signal Representations

- Binary number representations are sufficient for software programming languages

Binary			
Forcing 1	'1'	Forcing 0	'0'

- Physical wires cannot be modelled accurately using a binary number representation

- Additional values are necessary to accurately represent the state of a wire

Built-In Data Types

- VHDL supports a rich set of built-in data types as well as user-defined data types

Data Type	Characteristics
BIT	Binary, Unresolved
BIT_VECTOR	Binary, Unresolved, Array
INTEGER	Binary, Unresolved, Array
REAL	Floating Point

- Built-in data types work well for simulation but not so well for synthesis

- Built-in data types are suitable for use inside an architecture but should not be used for external pins

Logical Operators

- VHDL supports the following logical operators:

AND	NAND	NOT
OR	NOR	
XOR	XNOR	

- VHDL also supports the overloading of existing operators and the creation of new operators using functions

Other Operators

- VHDL supports the following relational operators:

  = (Equal)

  /= (Not Equal)

  < (Less Than)

  > (Greater Than)

- VHDL supports the following mathematical operators:

  + (Addition)

  - (Subtraction)

  * (Multiplication)

  / (Division)

Assignment Statements

```
SIGNAL a, b, c            : std_logic;
SIGNAL avec, bvec, cvec   : std_logic_vector(7 DOWNTO 0);

-- Concurrent Signal Assignment Statements
-- NOTE: Both a and avec are produced concurrently
a        <= b AND c;
avec     <= bvec OR cvec;

-- Alternatively, signals may be assigned constants
a        <= '0';
b        <= '1';
c        <= 'Z';
avec     <= "00111010";            -- Assigns 0x3A to avec
bvec     <= X"3A";                 -- Assigns 0x3A to bvec
cvec     <= X"3" & X"A";           -- Assigns 0x3A to cvec
```

Assignment Statements

```
SIGNAL a, b, c, d          :std_logic;
SIGNAL avec                :std_logic_vector(1 DOWNTO 0);
SIGNAL bvec                :std_logic_vector(2 DOWNTO 0);

-- Conditional Assignment Statement
-- NOTE: This implements a tree structure of logic gates
a <=     '0'       WHEN avec = "00" ELSE
         b         WHEN avec = "11" ELSE
         c         WHEN d = '1' ELSE
         '1';

-- Selected Signal Assignment Statement
-- NOTE: The selection values must be constants
bvec <=  d & avec;
WITH bvec SELECT
a <=     '0'       WHEN "000",
         b         WHEN "011",
         c         WHEN "1--",       -- Some tools won't synthesize '-' properly
         '1'       WHEN OTHERS;
```

Assignment Statements

```
SIGNAL a                    :std_logic;
SIGNAL avec, bvec           :std_logic_vector(7 DOWNTO 0);

-- Selected Signal Assignment Statement
-- NOTE: Selected signal assignments also work
--       with vectors
WITH a SELECT
avec <= "01010101"          WHEN '1',
        bvec                WHEN OTHERS;
```

Process Statements

- VHDL supports processes
- Processes encapsulate a portion of a design
- Processes have a sensitivity list that specifies signals and ports that cause changes in the outputs of the process
  - Sensitivity lists can be used to preserve the state of a hardware system
- For example, an edge-triggered flip-flop circuit is sensitive to a particular clock edge
  - The output of the edge-triggered flip-flop changes if and only if a particular clock edge is received
  - Otherwise, the previous output remains asserted

Process Statements

- The keywords used for conditional assignments and selected assignments differ from those used within a process:

Outside Processes	Inside Processes
`WHEN..ELSE`	`IF..ELSIF..ELSE..END IF`
`WITH..SELECT..WHEN`	`CASE..WHEN..END CASE`

- A selected assignment outside a process is functionally equivalent to a case statement within a process

- Processes can be used for combinational logic but most often, processes encapsulate sequential logic

Process Statements

```
SIGNAL reset, clock, d, q          :std_logic;

PROCESS (reset, clock)
-- reset and clock are in the sensitivity list to
-- indicate that they are important inputs to the process
BEGIN
   -- IF keyword is only valid in a process
   IF (reset = '0') THEN
        q <= 0;
   ELSIF (clock'EVENT AND clock = '1') THEN
        q <= d;
   END IF;
END PROCESS;
```

The EVENT attribute is true if an edge has been detected on the corresponding signal.

NOTE:

This implements a D flip-flop with an asynchronous active-low reset signal.

Process Statements

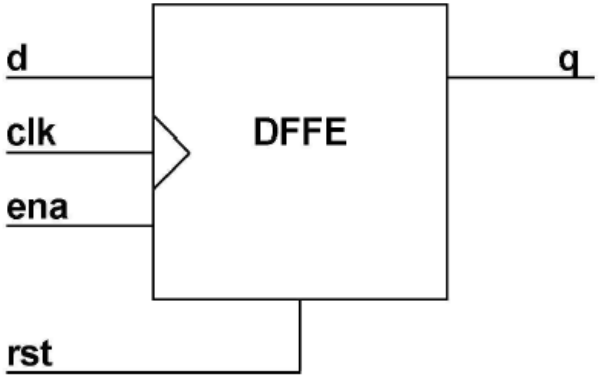
```
SIGNAL a, b, c, d          :std_logic;

PROCESS (a, b, d)
-- a, b, and d are in the sensitivity list to indicate that
-- the outputs of the process are sensitive to changes in them
BEGIN
   -- CASE keyword is only valid in a process
   CASE d IS
        WHEN '0' =>
                c <= a AND b;
        WHEN OTHERS =>
                c <= '1';
   END CASE;
END PROCESS;
```

NOTE:

This implements a combinational circuit.

D Flip-Flop Example

- Using a process and the `EVENT` attribute of a signal, it is possible to specify a D flip-flop
- The `EVENT` attribute can be used to check for the rising edge of a clock signal
- The block diagram of a D Flip-Flop is shown below:



D Flip-Flop Example

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY dffe IS
   PORT( rst, clk, ena, d : IN     std_logic;
         q                : OUT    std_logic );
END dffe;

ARCHITECTURE synthesis1 OF dffe IS
BEGIN
   PROCESS (rst, clk)
   BEGIN
         IF (rst = '1') THEN
                  q <= '0';
         ELSIF (clk'EVENT) AND (clk = '1') THEN
                  IF (ena = '1') THEN
                           q <= d;
                  END IF;
         END IF;
   END PROCESS;
END synthesis1;
```

Outline

- VHDL Examples
- VHDL Language
- **Advanced Examples**

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\encoder_using_if.vhd - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd

```vhdl
-----------------------------------------------------
-- Design Name : encoder_using_if
-- File Name   : encoder_using_if.vhd
-- Function    : Encoder using If
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity encoder_using_if is
    port (
        enable     :in  std_logic;                      --  Enable for the encoder
        encoder_in :in  std_logic_vector (15 downto 0); --  16-bit Input
        binary_out :out std_logic_vector (3 downto 0)   --  4 bit binary Output

    );
end entity;

architecture behavior of encoder_using_if is

begin
    process (enable, encoder_in) begin
        binary_out <= "0000";
        if (enable = '1') then
            if (encoder_in = X"0002") then binary_out <= "0001"; end if;
            if (encoder_in = X"0004") then binary_out <= "0010"; end if;
            if (encoder_in = X"0008") then binary_out <= "0011"; end if;
            if (encoder_in = X"0010") then binary_out <= "0100"; end if;
            if (encoder_in = X"0020") then binary_out <= "0101"; end if;
            if (encoder_in = X"0040") then binary_out <= "0110"; end if;
            if (encoder_in = X"0080") then binary_out <= "0111"; end if;
            if (encoder_in = X"0100") then binary_out <= "1000"; end if;
            if (encoder_in = X"0200") then binary_out <= "1001"; end if;
            if (encoder_in = X"0400") then binary_out <= "1010"; end if;
            if (encoder_in = X"0800") then binary_out <= "1011"; end if;
            if (encoder_in = X"1000") then binary_out <= "1100"; end if;
            if (encoder_in = X"2000") then binary_out <= "1101"; end if;
            if (encoder_in = X"4000") then binary_out <= "1110"; end if;
            if (encoder_in = X"8000") then binary_out <= "1111"; end if;
        end if;
    end process;
end architecture;
```

Line 5, Column 49

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\encoder_using_case.vhd • - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd × | encoder_using_case.vhd •

```vhdl
-----------------------------------------------------
-- Design Name : encoder_using_case
-- File Name   : encoder_using_case.vhd
-- Function    : Encoder using Case
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity encoder_using_case is
    port (
        enable     :in  std_logic;                      --  Enable for the encoder
        encoder_in :in  std_logic_vector (15 downto 0); --  16-bit Input
        binary_out :out std_logic_vector (3 downto 0)   --  4 bit binary Output

    );
end entity;

architecture behavior of encoder_using_case is

begin
    process (enable, encoder_in) begin
        if (enable = '1') then
            case (encoder_in) is
                when X"0002" => binary_out <= "0001";
                when X"0004" => binary_out <= "0010";
                when X"0008" => binary_out <= "0011";
                when X"0010" => binary_out <= "0100";
                when X"0020" => binary_out <= "0101";
                when X"0040" => binary_out <= "0110";
                when X"0080" => binary_out <= "0111";
                when X"0100" => binary_out <= "1000";
                when X"0200" => binary_out <= "1001";
                when X"0400" => binary_out <= "1010";
                when X"0800" => binary_out <= "1011";
                when X"1000" => binary_out <= "1100";
                when X"2000" => binary_out <= "1101";
                when X"4000" => binary_out <= "1110";
                when X"8000" => binary_out <= "1111";
                when others  => binary_out <= "0000";
            end case;
        end if;
    end process;
end architecture;
```

Line 4, Column 1

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\mux_using_with.vhd • - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd

```vhdl
-----------------------------------------------------
-- Design Name : mux_using_with
-- File Name   : mux_using_with.vhd
-- Function    : 2:1 Mux using with-select
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity mux_using_with is
    port (
        din_0   :in  std_logic; -- Mux first input
        din_1   :in  std_logic; -- Mux Second input
        sel     :in  std_logic; -- Select input
        mux_out :out std_logic  -- Mux output

    );
end entity;

architecture behavior of mux_using_with is

begin
    with (sel) select
    mux_out <= din_0 when '0',
               din_1 when others;

end architecture;
```

Line 6, Column 1

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\dlatch_reset.vhd - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd

```vhdl
-----------------------------------------------------
-- Design Name : dlatch_reset
-- File Name   : dlatch_reset.vhd
-- Function    : DLATCH async reset
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity dlatch_reset is
    port (
        data  :in  std_logic; -- Data input
        en    :in  std_logic; -- Enable input
        reset :in  std_logic; -- Reset input
        q     :out std_logic  -- Q output

    );
end entity;

architecture rtl of dlatch_reset is

begin
    process (en, reset, data) begin
        if (reset = '0') then
            q <= '0';
        elsif (en = '1') then
            q <= data;
        end if;
    end process;

end architecture;
```

Line 6, Column 1

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\dff_async_reset.vhd - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd | dff_async_reset.vhd

```vhdl
-----------------------------------------------------
-- Design Name : dff_async_reset
-- File Name   : dff_async_reset.vhd
-- Function    : D flip-flop async reset
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity dff_async_reset is
    port (
        data  :in  std_logic; -- Data input
        clk   :in  std_logic; -- Clock input
        reset :in  std_logic; -- Reset input
        q     :out std_logic  -- Q output

    );
end entity;

architecture rtl of dff_async_reset is

begin
    process (clk, reset) begin
        if (reset = '0') then
            q <= '0';
        elsif (rising_edge(clk)) then
            q <= data;
        end if;
    end process;

end architecture;
```

Line 6, Column 1; Saved D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\dff_async_reset.vhd (UTF-8)

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\dff_sync_reset.vhd - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd | dff_async_reset.vhd | dff_sync_reset.vhd

```vhdl
-----------------------------------------------------
-- Design Name : dff_sync_reset
-- File Name   : dff_sync_reset.vhd
-- Function    : D flip-flop sync reset
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity dff_sync_reset is
    port (
        data  :in  std_logic; -- Data input
        clk   :in  std_logic; -- Clock input
        reset :in  std_logic; -- Reset input
        q     :out std_logic  -- Q output

    );
end entity;

architecture rtl of dff_sync_reset is

begin
    process (clk) begin
        if (rising_edge(clk)) then
            if (reset = '0') then
                q <= '0';
            else
                q <= data;
            end if;
        end if;
    end process;

end architecture;
```

Line 22, Column 1

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\up_counter.vhd • - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd | dff_async_reset.vhd | dff_sync_reset.vhd | up_counter.vhd

```vhdl
-----------------------------------------------------
-- Design Name : up_counter
-- File Name   : up_counter.vhd
-- Function    : Up counter
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_unsigned.all;

entity up_counter is
    port (
        cout   :out std_logic_vector (7 downto 0);  -- Output of the counter
        enable :in  std_logic;                      -- Enable counting
        clk    :in  std_logic;                      -- Input clock
        reset  :in  std_logic                       -- Input reset
    );
end entity;

architecture rtl of up_counter is
    signal count :std_logic_vector (7 downto 0);
begin
    process (clk, reset) begin
        if (reset = '1') then
            count <= (others=>'0');
        elsif (rising_edge(clk)) then
            if (enable = '1') then
                count <= count + 1;
            end if;
        end if;
    end process;
    cout <= count;
end architecture;
```

Line 6, Column 1

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\gray_counter.vhd - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd | dff_async_reset.vhd | dff_sync_reset.vhd | up_counter.vhd | gray_counter.vhd

```vhdl
-----------------------------------------------------
-- Design Name : gray_counter
-- File Name   : gray_counter.vhd
-- Function    : 8 bit gray counters
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_unsigned.all;

entity gray_counter is
    port (
        cout   :out std_logic_vector (7 downto 0);  -- Output of the counter
        enable :in  std_logic;                      -- Enable counting
        clk    :in  std_logic;                      -- Input clock
        reset  :in  std_logic                       -- Input reset
    );
end entity;

architecture rtl of gray_counter is
    signal count :std_logic_vector (7 downto 0);
begin
    process (clk, reset) begin
        if (reset = '1') then
            count <= (others=>'0');
        elsif (rising_edge(clk)) then
            if (enable = '1') then
                count <= count + 1;
            end if;
        end if;
    end process;
    cout <= (count(7) &
            (count(7) xor count(6)) &
            (count(6) xor count(5)) &
            (count(5) xor count(4)) &
            (count(4) xor count(3)) &
            (count(3) xor count(2)) &
            (count(2) xor count(1)) &
            (count(1) xor count(0)) );
end architecture;
```

Line 6, Column 1

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\parity_using_assign.vhd • - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd | dff_async_reset.vhd | dff_sync_reset.vhd | up_counter.vhd | gray_counter.vhd | parity_using_assign.vhd

```vhdl
-----------------------------------------------------
-- Design Name : parity_using_assign
-- File Name   : parity_using_assign.vhd
-- Function    : Parity using direct assignment
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity parity_using_assign is
    port (
        data_in     :in  std_logic_vector (7 downto 0);
        parity_out  :out std_logic
    );
end entity;

architecture rtl of parity_using_assign is

begin

    parity_out <= (data_in(0) xor data_in(1)) xor
                  (data_in(2) xor data_in(3)) xor
                  (data_in(4) xor data_in(5)) xor
                  (data_in(6) xor data_in(7));
end architecture;
```

Line 6, Column 1

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\parity_using_function.vhd • - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd | dff_async_reset.vhd | dff_sync_reset.vhd | up_counter.vhd | gray_counter.vhd | parity_using_assign.vhd | parity_us

```vhdl
-----------------------------------------------------
-- Design Name : parity_using_function
-- File Name   : parity_using_function.vhd
-- Function    : Parity using function
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity parity_using_function is
    port (
        data_in     :in  std_logic_vector (7 downto 0);
        parity_out  :out std_logic
    );
end entity;

architecture rtl of parity_using_function is

    function parity(d_in :std_logic_vector) return std_logic is
        variable result :std_logic;
    begin
        result := (d_in(0) xor d_in(1)) xor
                  (d_in(2) xor d_in(3)) xor
                  (d_in(4) xor d_in(5)) xor
                  (d_in(6) xor d_in(7));

        return result;

    end parity;


begin
                  -- Function Call
    parity_out <= parity(data_in);

end architecture;
```

Line 5, Column 49

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\parity_using_function2.vhd - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_using_if.vhd | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd | dff_async_reset.vhd | dff_sync_reset.vhd | up_counter.vhd | gray_counter.vhd | parity_using_assign.vhd | parity_using_f

```vhdl
-----------------------------------------------------
-- Design Name : parity_using_function2
-- File Name   : parity_using_function2.vhd
-- Function    : Parity using function
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity parity_using_function2 is
    port (
        data_in     :in  std_logic_vector (31 downto 0);
        parity_out  :out std_logic
    );
end entity;

architecture rtl of parity_using_function2 is

    function parity (d_in :std_logic_vector) return std_logic is
        variable result :std_logic;
    begin
        result := '0';
        for i in 0 to d_in'length-1 loop
            result := result xor d_in(i);
        end loop;

        return result;

    end parity;


begin
                -- Function Call
    parity_out <= parity(data_in);

end architecture;

```

Line 6, Column 1; Saved D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\parity_using_function2.vhd (UTF-8)

Ask me anything

D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\parity_using_bitwise.vhd - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

encoder_us | encoder_using_case.vhd | mux_using_with.vhd | dlatch_reset.vhd | dff_async_reset.vhd | dff_sync_reset.vhd | up_counter.vhd | gray_counter.vhd | parity_using_assign.vhd | parity_using_function.vhd | parity_using

```vhdl
-----------------------------------------------------
-- Design Name : parity_using_bitwise
-- File Name   : parity_using_bitwise.vhd
-- Function    : Parity using bitwise xor
-- Prepared by : Wenye Li for EIE2050 at CUHK-SZ
-----------------------------------------------------
library ieee;
    use ieee.std_logic_1164.all;

entity parity_using_bitwise is
    port (
        data_in     :in  std_logic_vector (7 downto 0);
        parity_out  :out std_logic
    );
end entity;

architecture rtl of parity_using_bitwise is

begin

    process (data_in)
        variable pbit :std_logic;
    begin
        pbit := '0';
        for i in 0 to 7 loop
            pbit := pbit xor data_in(i);
        end loop;

        parity_out <= pbit;

    end process;

end architecture;
```

Line 6, Column 1; Saved D:\google-drive\cuhk.cn\2016-17-2nd\vhdl-examples\parity_using_bitwise.vhd (UTF-8)

Ask me anything

Thanks