EECS 140/141

Lab 2: Introduction to VHDL

09/02/2016

EECS 140/141 GTAs

KU SCHOOL OF ENGINEERING
The University of Kansas

VHDL

- VHDL-VHSIC Hardware Description Language
  - VHSIC-Very High Speed Integrated Circuit Program
- VHDL describes Hardware, generally digital logic
- VHDL describes **behavior**
  - That is, what function does the hardware compute
- VHDL describes hardware as **structure**
  - That is, how hardware components are connected
- VHDL is not a programming language
- VHDL is a standard
  - IEEE standard 1076
  - Developed in early 1980's
  - See: https://en.wikipedia.org/wiki/VHDL

Data Objects-Signals, Constants,& Variables “names you can use in VHDL”

- Data Object Names
  a. Any alphanumeric character and the ‘_’ underscore character
- Data Object Name **cannot** be:
  a. A VHDL Keyword
  b. Must begin with a letter, cannot end with ‘_’
  c. Cannot have two successive underscores
- SIGNAL Data Object:
  a. Syntax: [SIGNAL signal_name : type_name;]
  b. More in structural modeling lab
- CONSTANT
  a. Syntax: [CONSTANT constant_name: type_name:=“value”;]
- VARIABLE
  a. Syntax: [VARIABLE temp: INTEGER;]
- Data Object Values and Numbers
  a. Single bit: ‘0’ or ’1’,
  b. Multibit/Vector: “0001” (4 bit), “10001”(5 bit)
  c. “0001” is equivalent to ‘0’, ‘0’, ‘0’, ‘1’
  d. Decimals do not require quotes. E.g. 128,9,15 etc.

Data Types

- BIT and BIT_VECTOR

a.	SIGNAL x1	:BIT;	**E.g.** x1<='0';
b.	SIGNAL C	:BITVECTOR (1 TO 4);	**E.g.** C<="1101";
c.	SIGNAL Byte	:BIT_VECTOR (5 DOWNTO 0);	**E.g.** Byte<="01010";

- STD_LOGIC , STD_LOGIC _VECTOR Types
  - a. SIGNAL x1 :STD_LOGIC;
  - b. SIGNAL C :STD_LOGIC_VECTOR (1 TO 4);
  - c. SIGNAL Byte :STD_LOGIC_VECTOR (5 DOWNTO 0);
- SIGNED and UNSIGNED
- INTEGER
  - a. SIGNAL X: INTEGER RANGE -127 TO 127;
- BOOLEAN
  - a. SIGNAL Flag: Boolean;

Operators

- Exponentiation, Absolute value
  - ** ABS
- Multiplication, Division, Modulo, and Remainder
  - * / MOD REM
- Unary plus, and Unary minus
  - + -
- Addition, Subtraction, and Concatenation
  - + - &
- Relational Operators
  - == /= < <= > >=
- Logical Operators:
  - AND OR NAND NOR XOR XNOR NOT

VHDL Design Entity

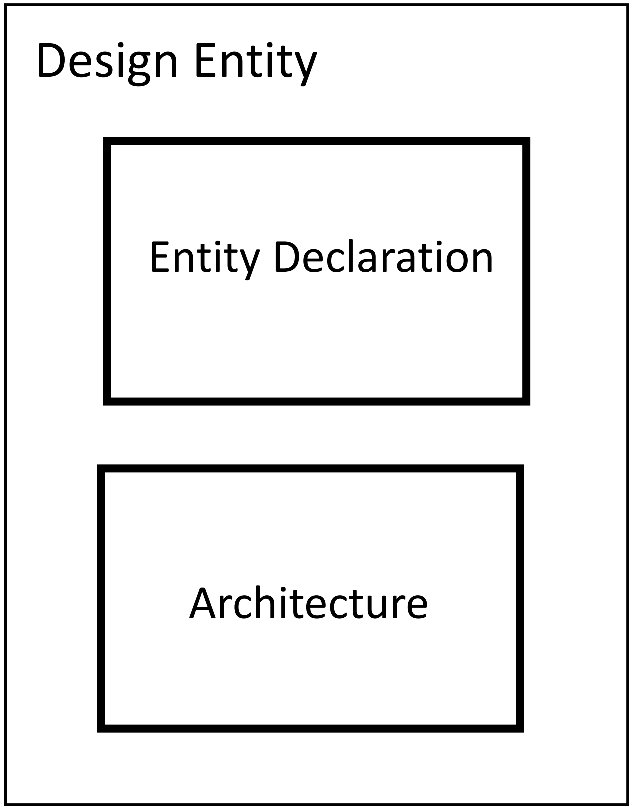
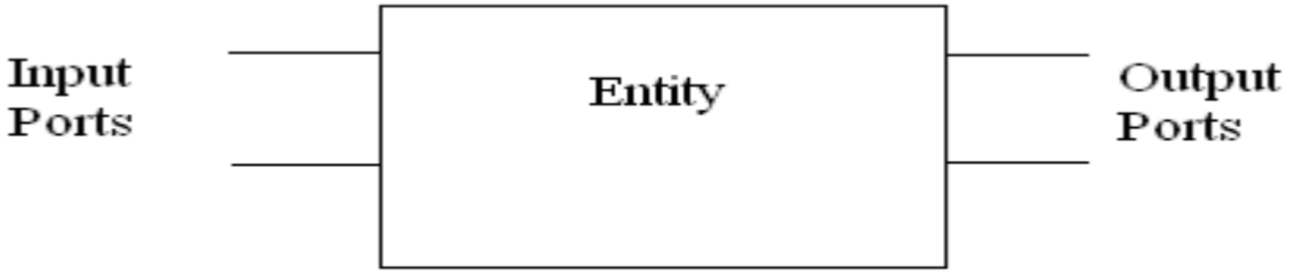


Figure 1 : The General Structure of VHDL Design Entity [EECS 140 Text Book]

VHDL Programming Structure

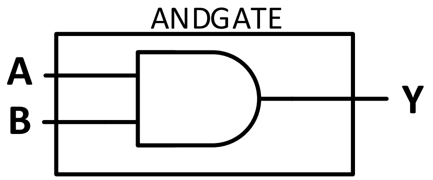
- VHDL Design
  - ❑ Circuit or sub-circuit described with VHDL code
    - a. VHDL Design has two main parts
      - i. Entity Declaration ii. Architecture
- Entity Declaration
  - ❑ Black Box view of a design



  - ❑ Specify the input and output signals that are interfaced
    - a. name, mode and type of input and output ports

VHDL Programming Structure

- Entity Syntax



```
Entity entity_name is
          Port ( signal_name1: mode type_name;
                 signal_name2: mode type_name;
                 signal_name3: mode type_name);
End Entity entity_name;
```

```
Entity ANDGATE is
          Port ( A: in std_logic;
                 B: in std_logic;
                 Y: out std_logic);
End Entity ANDGATE;
```

❑ user defined variables: black ink, keyword: colored ink

- Various <u>modes</u> for signals

In	signal input <u>to</u> an entity
Out	signal output <u>from</u> an entity
Inout	<u>both</u> input and output from an entity
Buffer	output signal, value <u>usable inside entity</u>

KU THE UNIVERSITY OF KANSAS

Rock Chalk, JAYHAWK!

VHDL Programming Structure

- Architecture
  - ❑ defines what is inside the Black Box
  - ❑ description of implementation of a design entity
  - ❑ each Entity has at least one Architecture
  - ❑ Design Techniques: Structural Model, Behavioral Model
- Architecture Syntax

```
Architecture architecture_name of entity_name is
        [architecture_declarations;]
Begin
      [ Statements;]
End Architecture architecture_name;
```

```
Architecture AND1 of ANDGATE is
      -- declarations
Begin
      -- statements
      Y <= A AND B;
 End Architecture AND1;
```

Behavioral Design/Modelling

- Directly specifies relationship between inputs and outputs
- Useful for simple logic networks with few gates
- The entire Boolean network or hardware is described inside the Architecture

Behavioral Example

- A description of input to output function

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

VHDL Package

```
ENTITY MajorityVote IS
      PORT( A, B, C, D, E:    IN    STD_LOGIC;
                  Q :         OUT   STD_LOGIC );
END MajorityVote;
```

ENTITY DECLARATION

```
ARCHITECTURE Behavior OF MajorityVote IS
      BEGIN
            Q <= ( B AND C AND E ) OR
                        ( C AND D AND E ) OR
                        ( B AND D AND E ) OR
                        ( B AND C AND D ) OR
                        ( A AND B AND C ) OR
                        ( A AND D AND E ) OR
                        ( A AND C AND E ) OR
                        ( A AND C AND D ) OR
                        ( A AND B AND E ) OR
                        ( A AND B AND D );
      END Behavior;
```

Direct relationship between inputs and output

Assignment Statements

- Two categories of assignment statements
  - ❑Concurrent (parallel execution of statements)
  - ❑Sequential (one after another; serial execution)
- Concurrent Assignment Statements
  - ❑Used to assign a value to a signal in architecture
  - ❑Four different types
    a. Simple signal assignment
    b. Selected signal assignment
    c. Conditional Signal assignment
    d. Generate Statements

Assignment Statements

- Concurrent Assignments
  - ❑Simple Signal Assignment: Used for logic or arithmetic expression

i. Syntax

```
Signal_name <= expression;
```

ii. Example of **Logical** expression

Declaration:

```
SIGNAL x1, x2, x3, f : STD_LOGIC;
```

Assignment:

```
f <= (x1 AND x2) OR x3;
```

iii. Example of **Arithmetic** expression

Declaration:

```
SIGNAL X,Y,S : STD_LOGIC_VECTOR (3 DOWNTO 0);
```

Assignment:

```
S <= X + Y;
```

Assignment Statements

- Concurrent Assignments
  - ❑Assigning SIGNAL Values Using OTHERS
  - ❑Used to set all bits in the signal S to a *value*
    1. Syntax

       S <= (OTHERS => *value*);
    2. Example- Set all bits in S to 0

       Declaration:

       ```
       SIGNAL  S : STD_LOGIC;
       ```

       Assignment:

       ```
       S <= (OTHERS => '0');
       ```

Assignment Statements WITH-SELECT-WHEN usage

- Concurrent Assignments
  - ❑Selected Signal Assignment
    a. Used to set the value of a signal to one of several alternatives based on a selection criterion
       i. Syntax

```
WITH expression SELECT
                signal_name <= object1 WHEN constant_value1,
                               object2 WHEN constant_value2;
```

       ii. Example: 2-to-1 multiplexer with *Sel* as the select input

Declaration:

```
                SIGNAL x1, x2, Sel, f  : STD_LOGIC;
```

Assignment:

```
                WITH Sel SELECT
                        f <= x1 WHEN '0',
                             x2 WHEN OTHERS;
```

       iii. Each criterion of WHEN clause must be mutually exclusive of the other criterion

Assignment Statements WHEN-ELSE usage

- Concurrent Assignments
  - ❑Conditional Signal Assignment
    a. Used to set a signal to one of several alternative values
       i. Syntax:

```
signal_name <= expression WHEN logic_expression ELSE
               {expression WHEN logic_expression ELSE}
               expression;
```

       ii. Example: Priority Encoder

```
f <= "01" WHEN req1 = '1' ELSE
     "10" WHEN req2 = '1' ELSE
     "11" WHEN req3 = '1' ELSE
     "00";
```

    b. The key difference between Selected and Conditional signal assignment is that the conditions listed in conditional signal assignment need not be mutually exclusive
    c. The conditions are given from the first listed to last listed

Assignment Statements

- Sequential Assignment Statements
  - ❑Used to assign a value to a signal in an process body
  - ❑The order in which the sequential statements appear in VHDL code affects the semantics of the code
  - ❑Three different types
    a. IF statement
    b. CASE statement
    c. LOOP statement

Assignment Statements

- Sequential Assignments
  - IF statement
    a. Condition is boolean expression
    b. Optional elsif sequence
       i. Conditions may overlap
       ii. Priority
    c. Optional else path
       i. Executed, if all conditions evaluate to false
    d. Syntax:

```
IF condition THEN
        statement;
        {statement;}
ELSIF condition THEN
        statement;
        {statement;}
ELSE
        statement;
        {statement;}
ENDIF;
```

Assignment Statements

- Sequential Assignments
  - ❑CASE statement
    a. WHEN clauses in CASE statement is mutually exclusive
    b. The keyword WHEN is used to identify constant values that the expression might match. The expression evaluates a choice, and then the associated statements will be executed
    c. The CASE statement will exit when all statements associated with the first matching constant value are executed
    d. Syntax:

```
CASE expression IS
        WHEN constant_value =>
                statement;
                {statement;}
        WHEN constant_value =>
                statement;
                {statement;}
        WHEN OTHERS =>
                statement;
                {statement;}
END CASE;
```

VHDL DEMO

- Demonstration by GTA
  - ❑Create new project in H:// drive (make EECS140 folder)
  - ❑Create new VHDL script using design entry wizard
  - ❑Write simple behavioral VHDL code to implement

Y = A'.B' + B.C' + B'.C

- Implement this design on the Basys3 board
  - ❑Create Xilinx Design Constraints file (XDC)
    a. Associate inputs and outputs of design to right pins on the FPGA
  - ❑Assign the inputs
    a. A, B and C to slide switches V17, V16, W16.
  - ❑Assign the output
    a. Use LED U16 to check output Y.