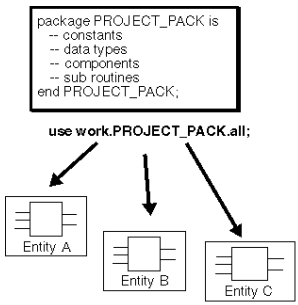
ECE 3401 Lecture 6

VHDL Behavioral Modeling & Concurrent Statements

VHDL Structural Elements

- **Entity:** description of interface consisting of the port list.
  - The primary hardware abstraction in VHDL, analogous to a symbol in a block diagram.
- **Architecture:** description of the function of the corresponding module.
- **Process:** allows for a sequential execution of the assignments
- **Configuration:** used for simulation purposes.
- **Package:** hold the definition of commonly used data types, constants and subprograms.
- **Library:** the logical name of a collection of compiled VHDL units (object code).
  - Mapped by the simulation or synthesis tools.

Packages Declaration



- Collection of definitions of constants, data types, components and subprograms,.
- A package serves as a central repository for frequently used utilities, such as component declarations.
- The declarations may then be reused by any VHDL model by simply accessing the package.
- **use WORK.LOGIC_OPS.all;**
  - The architecture accesses the component declarations in the package "LOGIC_OPS" located in the library WORK via the use clause.
  - Use clause placed just before the architecture body statement.

Example of Package Declaration

```
Package EXAMPLE_PACK is
    type SUMMER is (JUN, JUL, AUG);
    component D_FLIP_FLOP
        port (  D, CK     : in    BIT;
                Q, QBAR : out BIT);
    end component;
    constant PIN2PIN_DELAY : TIME := 125ns;
    function INT2BIT_VEC (INT_VALUE : INTEGER)
      return BIT_VECTOR;
end EXAMPLE_PACK

library DESIGN_LIB;                          -- library clause
use DESIGN_LIB.EXAMPLE_PACK.all;    -- use clause
entity  EXAM is  .....
```

Package Body

- **Package body** is used to store the definitions of functions and procedures that were declared in the corresponding package declaration.
- A package body is always associated with a package declaration.
- **Example:**

```
package body EXAMPLE_PACK is
   function INT2BIT_VEC (INT_VALUE: INTEGER)
         return BIT_VECTOR is
   begin
      -- behavior of function described here
   end INT2BIT_VEC;
 end EXAMPLE_PACK;
```

Example for Package

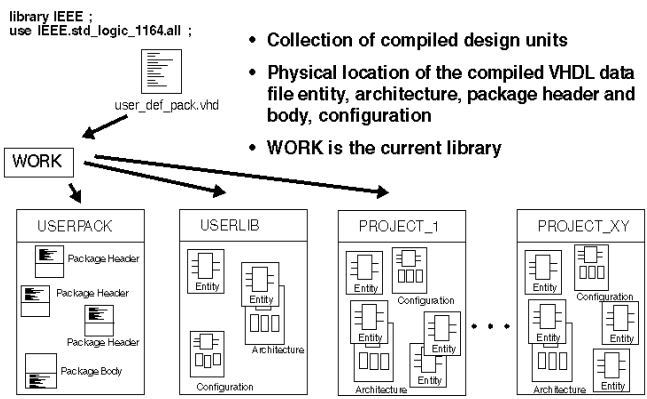
```
-- package
package LOGIC_OPS is
-- Declare logic operators
component AND2_OP
  port (A, B : in BIT;
             Z : out BIT);
 end component;

component OR3_OP
  port (A, B, C : in BIT;
          Z     : out BIT);
 end component;
```

```
component NOT_OP
  port ( A      : in  BIT;
         A_BAR : out BIT);
end component;

end LOGIC_OPS;
```

Example for Package Usage

```
-- Interface
entity MAJORITY is
    port ( A_IN, B_IN, C_IN : in BIT;
           Z_OUT            : out BIT);
end MAJORITY;

-- Body
-- use components in package LOGIC_OPS in library WORK
use WORK.LOGIC_OPS.all;
architecture STRUCTURE of MAJORITY is
    signal  INT1, INT2, INT3 : BIT;
begin
    A1: AND2_OP port map (A_IN, B_IN, INT1);
    A2: AND2_OP port map (A_IN, C_IN, INT2);
    A3: AND2_OP port map (B_IN, C_IN, INT3);
    O1: OR3_OP port map (INT1, INT2, INT3, Z_OUT);
end STRUCTURE;
```

Library (1)



- Collection of compiled design units
- Physical location of the compiled VHDL data file entity, architecture, package header and body, configuration
- WORK is the current library

Library (2)

- There are two reserved library names always available to designers.
  - **WORK**: Predefined library
  - **STD:** The **STD** contains two packages: **STANDARD** provides declaration for predefined types (real, integers, Boolean, etc). **TEXTIO** contains useful subprograms that enables to perform ASCII file manipulations
    - **Library** STD; -- declares STD to be a library
    - **use** STD.STANDARD.**all;** -- use all declarations in package STANDARD, such as BIT, located within the library STD.
- A Library cl*ause* declares **WORK** to be a library.
  Library WORK; -- WORK is predefined library

Library (3)

- Use-defined libraries: LOGIC_LIB
  - Assume the LOGIC_OPS package is located in the LOGIC_LIB library instead of WORK library.

```
library LOGIC_LIB;
use LOGIC_LIB.LOGIC_OPS.all;
architecture STRUCTURE of MAJORITY is
-- use components in package LOGIC_OPS in library
   LOGIC_LIB
```

VHDL Model

- Structural model: describe how it is composed of subsystems
  - Component declaration and instantiation
- Behavioral model: describe the function of entity in an abstract way
  - Algorithmic modeling (Procedural):
  - Data Flow modeling (Nonprocedural):
    - One kind of behavioral modeling.
    - Use concurrent statements.

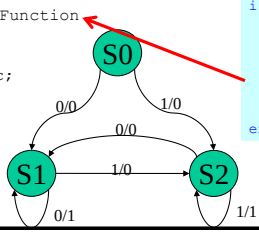
Algorithmic Model Example – State Diagram

```
architecture ALGORITHMIC of TWO_CON is
  type STATE is (S0, S1, S2);
  signal Q:STATE:=S0;
begin
  process (R, X, CLK, Q)
  begin
        Reset Event
        Clock Event
        Output Function
  end process;
end algorithmic;
```

```
if(R'EVENT and R='0') then
  Q<=S0;
```

```
elsif(CLK'EVENT and CLK='1') then
        if X='0' then
                Q<=S1;
        else
                Q<=S2;
        end if;
end if;
```

```
if(Q'EVENT or X'EVENT) then
        if(Q=S1 and X='0')or
          (Q=S2 and X='1') then
                Z<='1';
        else
                Z<='0';
        end if;
end if;
```



Data Flow Model –Truth Table

Let:

S0=>00

S1=>01

S2=>11

Y1/Y0 be the msb/lsb of state variable

Y1'=X

Y0'=1

Y1	Y0	X	Y1'	Y0'	Z
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	1	0
1	0	0	-	-	-
1	0	1	-	-	-
1	1	0	0	1	0
1	1	1	1	1	1

Z=Y0 and ((not Y1 and not X) or (Y1 and X))

Data Flow Model in VHDL

```
architecture DATAFLOW of TWO_CON is
  signal Y1, Y0:BIT;
begin
  STATE: block ((CLK='1' and CLK'EVENT) or R='0')
  begin
       Y1<=guarded '0' when R='0' else X;
       Y0<=guarded '0' when R='0' else 1;
  end block STATE;

  Z<=Y0 and ((not Y1 and not X) or (Y1 and X));
end DATAFLOW;
```

Classification

- VHDL provides several types of statements that can be used to assign logic values to signals.
  - **Concurrent Statements**
    - The term concurrent means that the VHDL statements are executed only when associated signals change value. There is no master procedural flow control, each concurrent statement executes when driven by an event.
  - **Sequential Statements**
    - Most statements found in programming languages such as BASIC, PASCAL, C, etc. execute in a sequential fashion. Sequential statements execute only when encountered by the procedural flow of control and the textual order in which statements appear determines the order in which they execute.

Concurrent Statements

- VHDL provides four different types of concurrent statements namely:
  - Signal Assignment Statement
    - Simple Assignment Statement
    - Selected Assignment Statement
    - Conditional Assignment Statement
  - Component Instantiation Statement
  - Generate Statement
  - Assert Statement

Simple Signal Assignment Statement

- A simple signal assignment statement is used for a logic or an arithmetic expression
- General format is:
  - Signal name <= Expression;
  - <= : VHDL assignment operator.
    - It is the only operator which can be used to assign a waveform to a signal.
- Example:

```
f <= (x1 AND x2) NOR (NOT x2 AND x3);
```

Simple Signal Assignment Statement

```
ENTITY example IS                                   -- line 1
       PORT (x1, x2, x3      : IN BIT;              -- line 2
                        f    : OUT BIT);            -- line 3
END example;                                        -- line 4

ARCHITECTURE LogicFunc OF example IS                -- line 5
BEGIN                                               -- line 6
      -- Simple signal assignment statement
       f<=(x1 AND x2)NOR(NOT x2 AND x3);            -- line 7
END LogicFunc;                                      -- line 8
```

Operators

- **Logical: not, and, or, nand, nor, xor, xnor**
- **Arithmetic:**

Operator	Definition
+	addition
-	Subtraction
*	Multiply
/	divide
**	Exponentiation
MOD	modulus
REM	remainder
&	Concatenation

- **Relational:**

Operator	Definition
=	equal
/=	not equal
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal

Miscellaneous Operators

Operator	Definition
ABS	Absolute Value
SLL	Shift left logical
SRL	Shift right logical
SLA	Shift left arithmetic
SRA	Shift right arithmetic
ROL	Rotate left
ROR	Rotate right

4-bit Adder Example

```
LIBRARY ieee;                                          -- line 1
USE  ieee.std_logic_1164.all;                          -- line 2
USE  ieee.std_logic_arith.all;                         -- line 3
USE  ieee.std_logic_signed.all;                        -- line 4

ENTITY adder IS                                              -- line 5
PORT (Cin     : IN  STD_LOGIC;                         -- line 6
      a, b    : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);      -- line 7
      s : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);-- line 8
      Cout    : OUT STD_LOGIC);                        -- line 9
END adder;                                                  -- line 10

ARCHITECTURE example OF adder IS                       -- line 11
 SIGNAL Sum : STD_LOGIC_VECTOR(4 DOWNTO 0);            -- line 12
BEGIN                                                        -- line 13
        Sum <= a + b + Cin;                                  -- line 14
        s <= Sum(3 DOWNTO 0);                          -- line 15
        Cout <= Sum(4);                                -- line 16
END example;                                           -- line 17
```

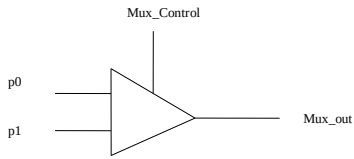
Selected Signal Assignment Statement

- Selected signal assignment statement is used to set the values of a signal to one of several alternatives based on a selection criterion.
- Format:

```
WITH discriminant SELECT
target_signal <=   value1 WHEN choice1,
                   value2 WHEN choice2,
                            .
                            .
                   valueN WHEN choiceN [or OTHERS];
```

  - Note: N choices controlled by signal discriminant.

MUX Example



Write VHDL code for the behavior of the entity using selected assignment statement, such that Mux_Out is assigned the value of p0 when Mux_Control = 0; otherwise, Mux_Out is assigned the value p1.

MUX VHDL

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux2to1 IS
      PORT (p0, p1, Mux_Control :      IN  STD_LOGIC;
           Mux_Out :     OUT  STD_LOGIC);
  END mux2to1;

ARCHITECTURE arch1 OF mux2to1 IS
BEGIN
      WITH Mux_Control SELECT
      Mux_Out <=   p0 WHEN  '0' ,
                p1 WHEN OTHERS;
END arch1;
```

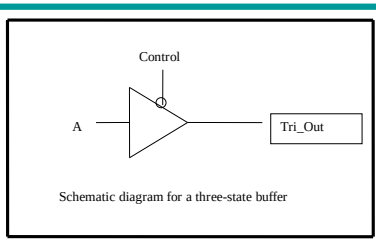
Conditional Assignment Statement

- A conditional signal assignment allows a signal to be set to one of several alternative values.
- The format is

```
target_signal <= value1 WHEN condition1 ELSE
                 value2 WHEN condition2 ELSE
                        …
                 valueN-1 WHEN conditionN-1 ELSE
                 valueN;
```

  * Note: executes in the textual order.
- The conditions must be Boolean expressions (‘0’ or ‘1’ only).

  For example: (N='1' and SEL= '0')
  - If condition is found to be true, the corresponding expression is evaluated and the resulting value is assigned to the target signal.
  - If none of the conditions are found to be true, the value valueN is assigned to the target signal.



Schematic diagram for a three-state buffer

- **Example:** a three-state logic gate.
  - Write a VHDL code for the behavior of the entity using conditional assignment statement, such that Tri_out is assigned the value of A when Tri_control = 0; otherwise, Tri_out is assigned the value Z.

TriState Example

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

ENTITY tristate IS
  PORT (A, Control: IN STD_LOGIC;
        Tri_Out   : OUT STD_LOGIC);
END tristate;

ARCHITECTURE data_flow OF tristate IS
BEGIN
  Tri_Out <= A    WHEN Control = '0'  ELSE
             'Z' ;
END data_flow;
```

Example: 4-to-2 Priority Encoder

- The output of a **priority encoder** indicates the active input that has the highest priority. When an input with the higher priority is asserted, the other inputs with the lower priority are ignored.
- Assume that d0 has the lowest priority and d3 the highest priority.

d0 has the lowest priority and d3 the highest priority

D3	D2	D1	D0	F1	F2	Z
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

The outputs F1 and F2 represent the binary number that identifies the highest priority input set to 1. Output Z is set to 1 if at least one of the inputs is one. It is set to zero when all inputs is zero.

Priority Encoder VHDL

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

ENTITY priority IS
   PORT (d : IN     STD_LOGIC VECTOR (3 DOWNTO 0);
         f : OUT    STD_LOGIC VECTOR (1 DOWNTO 0);
         z :        OUT    STD_LOGIC);
END priority;

ARCHITECTURE arch_cond OF priority IS
BEGIN
     f <= "11" WHEN d(3) =  '1'  ELSE
          "10" WHEN d(2) =  '1'  ELSE
          "01" WHEN d(1) =  '1'  ELSE
          "00";
      z <=  '0'  WHEN d = "0000" ELSE  '1'   ;
END arch_cond;
```