

# VHDL 2 Slides

# Data Types and Operators

Operator Class	Operators	BOOLEAN	BIT	BIT_VECTOR	STD_LOGIC	STD_LOGIC_VECTOR	CHARACTER	STRING	SEVERITY_LEVEL	INTEGER	REAL	TIME
Logical	AND, OR, NAND, NOR, XOR, XNOR	X	X	X	X	X						
Other	NOT	X	X	X	X	X			X			
Relational	=, /=, <, <=, >, >=	X	X	X	X	X	X	X		X	X	X
Shift	SLL, SRL, SLA, SRA, ROL, ROR			X								
Adding	+, -									X	X	X
Adding	&							X				
Sign	+, -									X	X	X
Multiplying	*, /									X	X	X
Multiplying	MOD, REM									X		
Other	**									X	X	
Other	ABS									X	X	X

Value	Description
U	Uninitialized
X	Forcing Unknown
0	Forcing 0
1	Forcing 1
Z	High Impedance
W	Weak Unknown
L	Weak 0
H	Weak 1
-	Don't Care

***IMPORTANT!***  
***SOME DATA TYPES AND VALUES ARE ONLY USED FOR SIMULATION!***

# Representing the Values

- Decimal Values - 7000 or 7e3 or 7E3
- BITS/Characters – ‘0’, ‘1’, ‘\$’, ‘X’...
- BIT\_VECTORS/STRINGS – (downto/to)
  - “00011010” vs 00011010 vs ‘1’
- Hexadecimal – x”1A”
- Examples

# Concurrent vs Sequential Code

	Statement	Description
Concurrent	BLOCK	An internal block representing a portion of a design
	X PROCESS	An independent sequential process representing the behavior of some portion of the design.
	CONCURRENT_PROCEDURE_CALL	A process containing the corresponding sequential procedure call statement.
	CONCURRENT_ASSERTION	A passive process statement containing the specified assertion statement.
	X CONCURRENT_SIGNAL_ASSIGNMENT	An equivalent process statement that assigns values to signals. Conditional or Selected <= (sequential assignment, if, case, or null)
	X COMPONENT_INSTANTIATION	Defines a subcomponent of the design entity in which it appears, associates signals or values with the ports of that subcomponent, and associates values with generics of that subcomponent.
	X GENERATE	Provides a mechanism for iterative or conditional elaboration of a portion of a description.
?	WAIT	Causes the suspension of a process statement or a procedure.
Sequential	ASSERTION	Checks that a specified condition is true and reports an error if it is not.
	REPORT	Displays a message.
	X SIGNAL_ASSIGNMENT	Modifies the projected output waveforms contained in the drivers of one or more signals. (pg. 115)
	X VARIABLE_ASSIGNMENT	Replaces the current value of a variable with a new value specified by an expression.
	PROCEDURE_CALL	Invokes the execution of a procedure body.
	X IF	Selects for execution one or none of the enclosed sequences of statements, depending on the value of one or more corresponding conditions
	X CASE	Selects for execution one of a number of alternative sequences of statements; the chosen alternative is defined by the value of an expression.
	X LOOP	Includes a sequence of statements that is to be executed repeatedly, zero or more times.
	X NEXT	Used to complete the execution of one of the iterations of an enclosing loop statement.
	X EXIT	Used to complete the execution of an enclosing loop statement.
	RETURN	Used to complete the execution of the innermost enclosing function or procedure body.
	NULL	Performs no action.

# Conditional Signal Assignments

## WHEN/ELSE

```
ARCHITECTURE dataflow OF mux_4_1_WE IS
BEGIN

    -- Implement 4 to 1 mux

    mux_out <= f0 WHEN (s1 = '0' AND s0 = '0') ELSE
               f1 WHEN (s1 = '0' AND s0 = '1') ELSE
               f2 WHEN (s1 = '1' AND s0 = '0') ELSE
               f3;

END dataflow ;
```

## IF/THEN/ELSIF/THEN/END IF (Must be inside a process statement)

```
ARCHITECTURE ifarc OF mux IS
BEGIN
    PROCESS
    BEGIN
        IF sel="00" THEN
            y<=x0;
        ELSIF sel="01" THEN
            y<=x1;
        ELSIF sel="10" THEN
            y<=x2;
        ELSIF sel="11" THEN
            Y<=x3;
        END IF;
    END PROCESS;
END ifarc;
```

# Selected Signal Assignments

## WITH/SELECT

```
ARCHITECTURE dataflow OF mux_4_1_WSW IS
BEGIN

    -- Implement 4 to 1 mux
    WITH s SELECT
        mux_out <= f0 WHEN "00",    -- ", " instead of ";"
                f1 WHEN "01",
                f2 WHEN "10",
                f3 WHEN OTHERS; -- cannot be " f3 WHEN "11" "

END dataflow ;
```

## CASE/IS/WHEN/END CASE (Must be inside a process statement)

```
ARCHITECTURE casearc OF mux IS
BEGIN
    PROCESS
    BEGIN
        CASE sel IS
            WHEN "00" =>    y<=x0;
            WHEN "01" =>    y<=x1;
            WHEN "10" =>    y<=x2;
            WHEN OTHERS =>  Y<=x3;
        END CASE;
    END PROCESS;
END casearc;
```

# Design Entity Instantiation

```
Instantiation_label : ENTITY WORK.entity_name(architecture) PORT MAP (  
    entity_signal_1=> top_level_signal_1,  
    entity_signal_2=> top_level_signal_2  
);
```

# Example

---

```
library work;
use work.all;

entity muxtop is
  PORT(
    sw : IN BIT_VECTOR (17 DOWNTO 0);
    ledr : OUT BIT_VECTOR (17 DOWNTO 0);
    ledg : OUT BIT_VECTOR (8 DOWNTO 0)
  );
END muxtop;

ARCHITECTURE arc OF muxtop IS
BEGIN
  u1: entity work.mux(arc) port map(
    x3=>sw(3),
    x2=>sw(2),
    x1=>sw(1),
    x0=>sw(0),
    sel=>sw(17 downto 16),
    y=>ledg(8)
  );
  ledr<=sw;
end;
```