VHDL Data Types

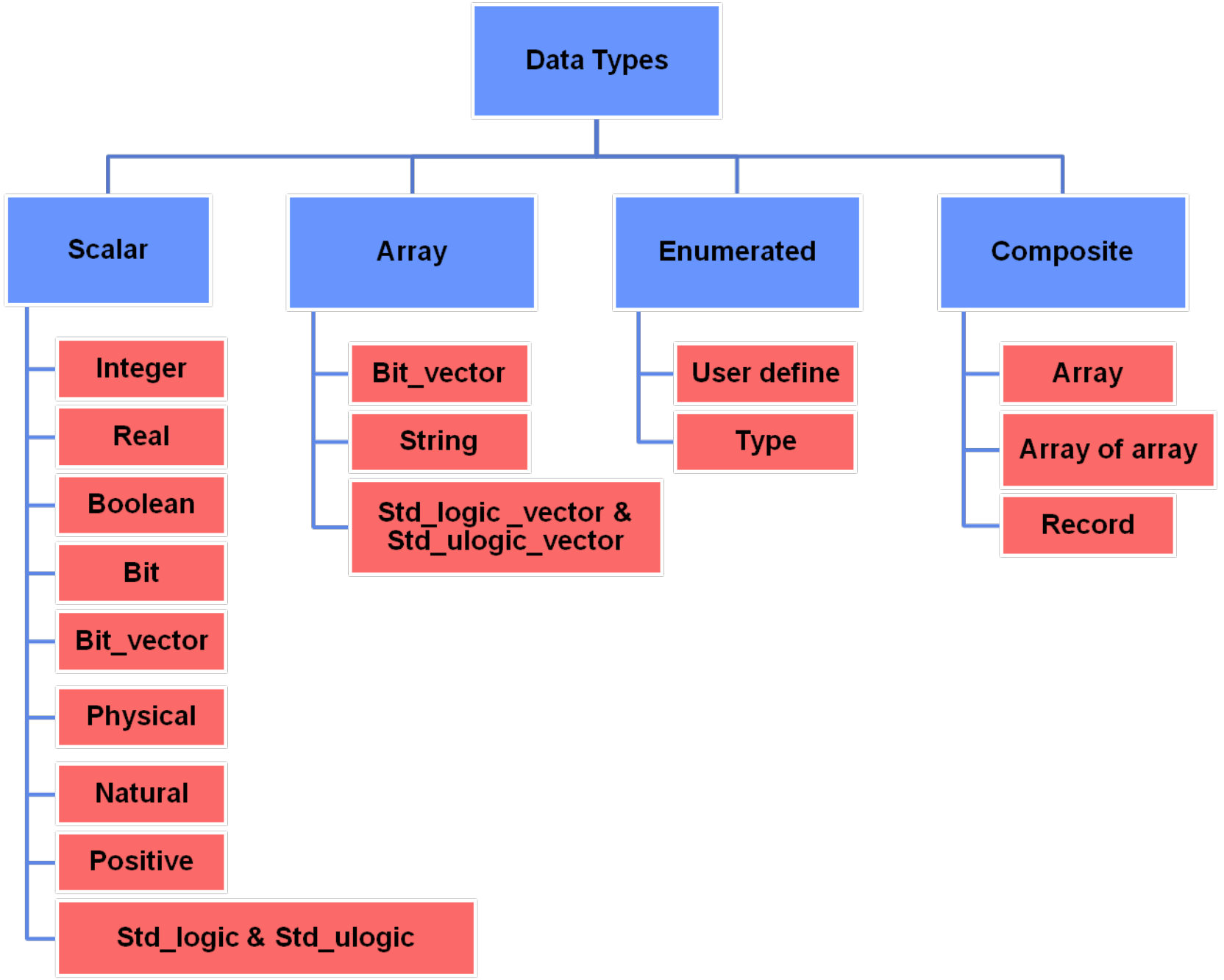
VHDL Data Types

- What is a “Data Type”?
  - This is a classification objects/items/data that defines the possible set of values which the objects/items/data belonging to that type may assume.
  - E.g. (VHDL) integer, bit, std_logic, std_logic_vector
  - Other languages (float, double, int , char etc)

VHDL Data Types

- Predefined Data Types
  - Specified through the IEEE 1076 and IEEE 1164 standards
  - The IEEE Standard 1076 defines the VHSIC Hardware Description Language or VHDL
    - Developed by Intermetrics, IBM and Texas Instruments for United States Air Force.
    - 1076-1987 was the first version
    - Revised in 1993, 2000, 2002, and 2008

VHDL Data Types



VHDL Data Types

- **Package standard of library std (Included by default):**
  - bit type (0, 1)
  - bit vectors (group of multi-bit signal → bus)
  - Example
    - SIGNAL x: BIT;
    - SIGNAL y: BIT_VECTOR (3 DOWNTO 0);
    - SIGNAL w: BIT_VECTOR (0 TO 7);
    -
  - Signal assignment operator **<=**
    - x <= '1';
    - y <= "0111";
    - w <= "01110001";

VHDL Data Types

- **Package standard of library std (Included by default):**
  - BOOLEAN (TRUE, FALSE)
    - Example
      - variable VAR1: boolean := FALSE;
  - INTEGER (32 bit, -2,147,483,647 to +2,147,483,647
    - Example
      - SIGNAL SUM: integer range 0 to 256 :=16;
  - REAL (from -1.0E38 to +1.0E38)
    - Example
      - constant Pi : real := 3.14159;

VHDL Data Types

- The IEEE Standard 1164
  - Introduce Multivalue Logic (std_logic_1164) Packages
  - The primary data type std_ulogic (standard unresolved logic) consists of nine character literals in the following order:

1. **'U' – uninitialized (default value)**
2. **'X' - strong drive, unknown logic value**
3. **'0' - strong drive, logic zero**
4. **'1' - strong drive, logic one**
5. **'Z' - high impedance (for tri-state logic)**
6. **'W' - weak drive, unknown logic value**
7. **'L' - weak drive, logic zero**
8. **'H' - weak drive, logic one**
9. **'-' - don't care**

- std_ulogic and its subtype (std_logic, std_logic_vector, std_ulogic_vector) values can be categorized in terms of their state and strength (forcing, weak and high impedance.)
- Weak strength is used for multi-driver inputs catering for pullup/pulldown

VHDL Data Types

- std_ulogic data type possible values and corresponding strength

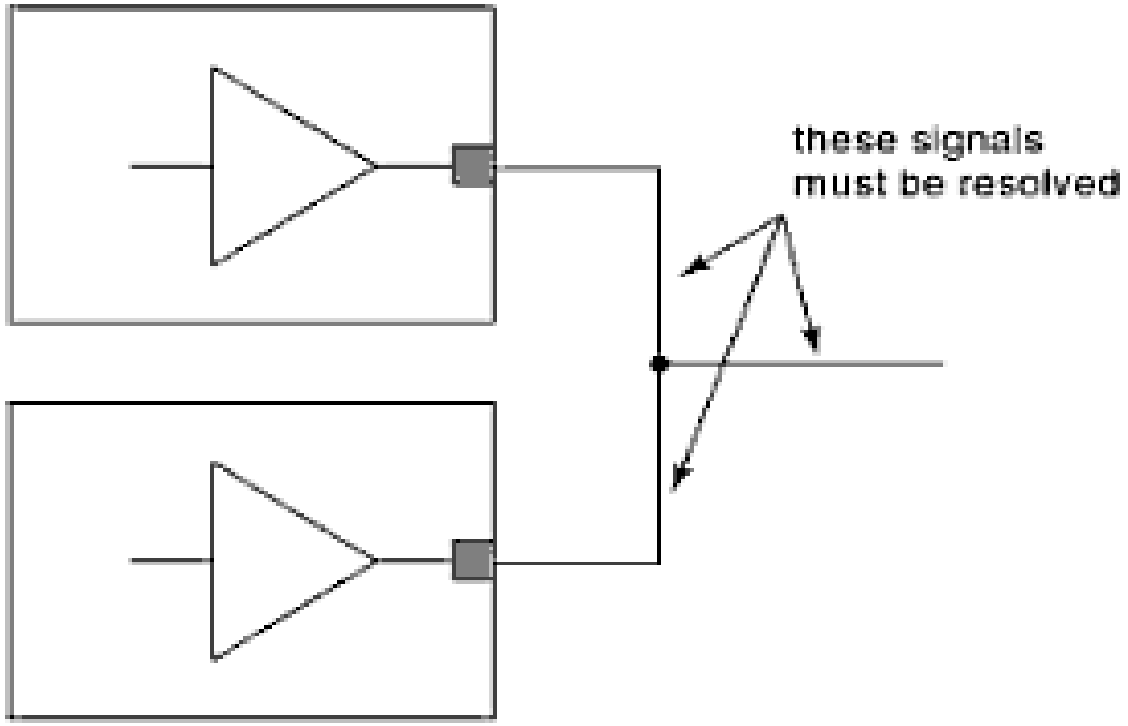
Data Value	State	Strength	Comment
U	Unitialised	None	Default value before simulation.
X	Unknown	Forcing	Represents driven signals whose value cannot be determined as 1 or 0
0	0	Forcing	Represents signals from active output drivers
1	1	Forcing	
Z	None	High Impedance	Represents output of tri-state buffer when not enabled.
W	Unknown	Weak	Represents signals from resistive drivers e.g. pull-up and pull-down resistors
L	0	Weak	
H	1	Weak	
-	Don't care	None	Allows synthesiser to decide whether to assign a 0 or a 1 for minimum systhesised logic circuit.

VHDL Data Types

- std_ulogic
  - Is an unresolved data type
  - Declared in package STD_LOGIC_1164 of library IEEE.
  - All data signals are of unresolved type by default.
  - Unresolved data type signals cannot be driven by more than one driver/sources. (adding multiples sources will result in compiler error).
  - Helps checking that designer has not accidentally assigned two sources to a signal.

VHDL Data Types

- Resolved Data Types
  - Always declared with a resolution function (within its library).
  - Resolution function defines all possible combinations of one or more source values and the correspond resolved value (result).



VHDL Data Types

- std_logic (this is a resolved data type)
  - A subtype of std_ulogic
  - Declared in package STD_LOGIC_1164 of library IEEE as

    subtype std_logic is resolved std_ulogic;
  - Specified a resolution function called “resolved”

VHDL Data Types

- std_logic resolution table

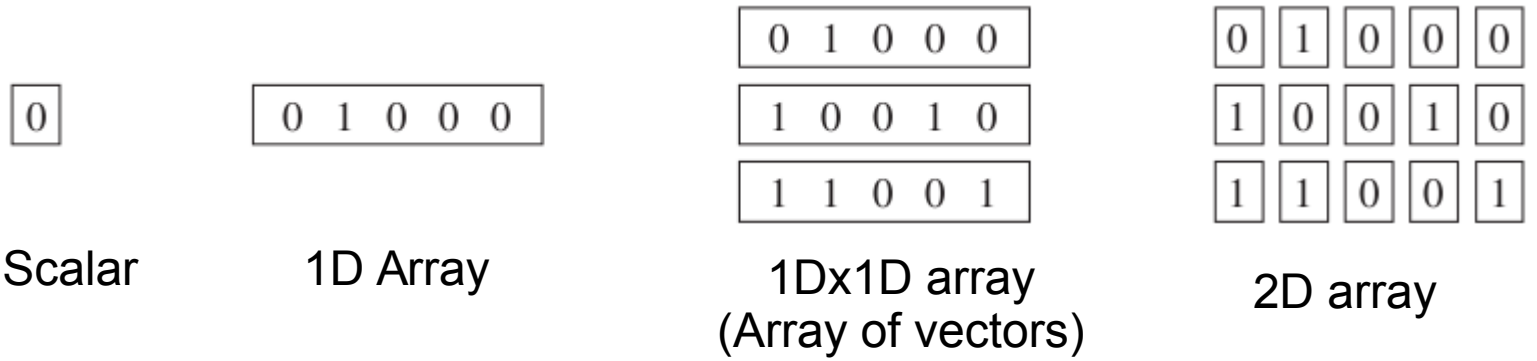
		X	0	1	Z	W	L	H	-
X		X	X	X	X	X	X	X	X
0		X	0	X	0	0	0	0	X
1		X	X	1	1	1	1	1	X
Z		X	0	1	Z	W	L	H	X
W		X	0	1	W	W	W	W	X
L		X	0	1	L	W	L	W	X
H		X	0	1	H	W	W	H	X
-		X	X	X	X	X	X	X	X

VHDL Data Types

- **std_logic declaration examples**
  - SIGNAL x: STD_LOGIC;
  - SIGNAL y: STD_LOGIC_VECTOR (3 DOWNTO 0) := "0001";

VHDL Data Types: Arrays

- Arrays are collections of objects of the same type.
- Can be 1D (1 dimensional) of 2D (2 dimensional) arrays.
- Higher dimensional arrays are not synthesizable



- There are no pre-defined 2D or 1Dx1D arrays; have to be defined by designer.

VHDL Data Types: Arrays

- **Defining VHDL Arrays**
  - First define a new data type
  - Second declare a signal, variable or constant of the defined data type.
- **General Format of Array definition**

  TYPE type_name IS ARRAY (specification) OF data_type;

  SIGNAL signal_name: type_name [:= initial_value];

VHDL Data Types: Arrays

- **Example:**

  TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;

  - Defines a row (1D array) (data type) with of seven STD_LOGIC values with MSB on left.

  TYPE matrix IS ARRAY (0 TO 3) OF row;

  - Defines an 1Dx1D ARRAY (matrix) data type containing 4 row defined in previous line.

  SIGNAL x: matrix;

  - Defines 1Dx1D signal of type matrix as defined in previous line

VHDL Data Types: Arrays

- **Example:1Dx1D Array (of vectors) --- Alternative method**

  TYPE matrix IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);

- **Example:2D Array Data type**

  TYPE matrix2D IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;

  - Array construction is not based on vectors, but rather entirely on scalars.
  - It is a 2 dimensional array of scalars

VHDL Data Types: Array Assignments

- **Type Definition:**

  TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC; -- 1D array
  TYPE array1 IS ARRAY (0 TO 3) OF row; -- 1Dx1D array

- **Signal Declaration;**

  SIGNAL x: row;
  SIGNAL y: array1;

- **Scalar Signal (array) assignment:**

  x(0) <= y(1)(2);

  - Note the two pairs of parentheses since y is a 1Dx1D array.

VHDL Data Types: Array Assignments

- **Type Definition:**

  TYPE array2 IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);

  -- 1Dx1D

  TYPE array3 IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;

  -- 2D array

- **Signal Declarations:**

  SIGNAL v: array2;

  SIGNAL w: array3;

- **Scalar Signal Assignments:**

  x(1) <= v(2)(3);

  x(2) <= w(2,1);

  - Single pair of parentheses since w is 2D array

VHDL Data Types: Array Assignments

```
TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;
TYPE array1 IS ARRAY (0 TO 3) OF row;
TYPE array2 IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
TYPE array3 IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;
```

- **Signal Declaration;**

```
SIGNAL x: row;
SIGNAL y: array1;
SIGNAL v: array2;
SIGNAL w: array3;
```

•Scalar Signal Assignments:

```
y(1)(1) <= x(6);
y(2)(0) <= v(0)(0);
y(0)(0) <= w(3,3);
w(1,1) <= x(7);
w(3,0) <= v(0)(3);
```

VHDL Data Types: Array Assignments

- Vector Signal Assignments

  TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;
  TYPE array1 IS ARRAY (0 TO 3) OF row;
  TYPE array2 IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
  TYPE array3 IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;

- **Signal Declaration;**

  SIGNAL x: row;
  SIGNAL y: array1;
  SIGNAL v: array2;
  SIGNAL w: array3;

- **Legal Assignments**

  x <= y(0);
  y(1)(7 DOWNTO 3) <= x(4 DOWNTO 0);
  v(1)(7 DOWNTO 3) <= v(2)(4 DOWNTO 0);

VHDL Data Types: Array Assignments

- **Vector Signal Assignments**

  TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;
  TYPE array1 IS ARRAY (0 TO 3) OF row;
  TYPE array2 IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
  TYPE array3 IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;

- **Signal Declaration;**

  SIGNAL x: row;
  SIGNAL y: array1;
  SIGNAL v: array2;
  SIGNAL w: array3;

- **Why are the following assignments illegal ?**

  x <= v(1);
  x <= w(2);
  x <= w(2, 2 DOWNTO 0);
  v(0) <= w(2, 2 DOWNTO 0);
  v(0) <= w(2);
  y(1) <= v(3);
  w(1, 5 DOWNTO 1) <= v(2)(4 DOWNTO 0);

VHDL OPERATORS

- Logical operators

Logical operation	Operator	Example
AND	AND	Z <= (A AND B);
NAND	NAND	Z <= (A NAND B);
NOR	NOR	Z <= (A NOR B);
NOT	NOT	Z <= NOT (A);
OR	OR	Z <= (A OR B);
XNOR	XNOR	Z <= (A XNOR B);
XOR	XOR	Z <= (A XOR B);

VHDL OPERATORS

- Arithmetic operators

Arithmetic operation	Operator	Example
Addition	+	`Z <= A + B;`
Subtraction	-	`Z <= A - B;`
Multiplication	*	`Z <= A * B;`
Division	/	`Z <= A / B;`
Exponentiating	**	`Z <= 4 ** 2;`
Modulus	MOD	`Z <= A MOD B;`
Remainder	REM	`Z <= A REM B;`
Absolute value	ABS	`Z <= ABS A;`

VHDL OPERATORS

- Relational operators

Relational operation	Operator	Example
Equal to	=	`If (A = B) Then`
Not equal to	/=	`If (A /= B) Then`
Less than	<	`If (A < B) Then`
Less than or equal to	<=	`If (A <= B) Then`
Greater than	>	`If (A > B) Then`
Greater than or equal to	>=	`If (A >= B) Then`

VHDL Reserved Words

- Reserved words cannot be used by designers for identifiers such as variables, signal names, etc.

abs	file	of	then
after	for	open	to
all		or	transport
and	generic	others	type
architecture		out	
array	if		until
	in	package	use
begin	inertial	port	
	inout	process	variable
case	is		
component		rem	wait
configuration	library	report	when
constant	linkage	rol	while
	loop	ror	with
downto			
	mod	select	xnor
else		signal	xor
elsif	nand	sla	
end	next	sll	
entity	nor	sra	
	not	srl	

Data Types:
Advanced Topics

VHDL Data Types

- Package std_logic_arith of library IEEE:
  - Defines SIGNED and UNSIGNED data types, plus several data conversion functions, like:
    - conv_integer(p),
    - conv_unsigned(p, b),
    - conv_signed(p, b), and
    - conv_std_logic_vector(p, b).
  - Allow arithmetic operations
  - Data conversion to be discussed in later slides

VHDL Data Types

- Packages std_logic_signed and std_logic_unsigned of library IEEE:
  - Contain functions that allow operations with STD_LOGIC_VECTOR data to be performed as if the data were of type SIGNED or UNSIGNED, respectively.

User Defined VHDL Data Types

- User Defined Integer Data Types
  - Subtype of Integer
  - Examples
    - TYPE integer IS RANGE -2147483647 TO +2147483647;
    - TYPE my_integer IS RANGE -32 TO 32;
    - -- A user-defined subset of integers.
    - TYPE student_grade IS RANGE 0 TO 100;
    - -- A user-defined subset of integers or naturals.
    - TYPE natural IS RANGE 0 TO +2147483647;

User Defined VHDL Data Types

- User Defined ENUMERATED Data Types
  - Data type consisting of a set of named values.
  - Examples
    - **TYPE bit IS ('0', '1');**

    - **TYPE my_logic IS ('0', '1', 'Z');**
      – This is the pre-defined type BIT

    - **TYPE bit_vector IS ARRAY (NATURAL RANGE <>) OF BIT;**
      -- This is the pre-defined type BIT_VECTOR.
      NATURAL RANGE <>, on the other hand, indicates that the only restriction is that the range must fall within the NATURAL range.

User Defined VHDL Data Types

- User Defined ENUMERATED Data Types
  - More Examples
    - TYPE state IS (idle, forward, backward, stop);
      -- An enumerated data type, typical of finite state machines.
      - Two bits will be used to encode this data type values.
      - Idle will be the default value

    - TYPE color IS (red, green, blue, white, black);
      -- Another enumerated data type.
      - Three bits will be used for encoding this data type.
      - Red will be the default value

VHDL Data Types:Records

- Like Arrays Arrays records are collections of objects.
- Unlike arrays records can contain objects of different data types.
- Example

```
TYPE birthday IS RECORD
    day: INTEGER RANGE 1 TO 31;
    month: month_name; – month_name datatype should be pre-
    defined
END RECORD;
```

VHDL Data Types:
Signed and Unsigned Types

- Defined in the **STD_LOGIC_ARITH** package of the IEEE library
- **For arithmetic operations.**
- Signal Declaration Examples

  SIGNAL x: SIGNED (7 DOWNTO 0);

  SIGNAL y: UNSIGNED (0 TO 3);
- Syntax is similar to that of STD_LOGIC_VECTOR not like integers
- An UNSIGNED value is a number never lower than zero.

  For example,
  - Unsigned ‘‘0101’’ = the decimal 5
  - Unsigned ‘‘1101’’ signifies 13.
  - Signed ‘‘0101’’ = the decimal 5
  - Signed ‘‘1101’’ signifies -3 (Two's complement)

VHDL Data Types:
Signed and Unsigned Types

- **Operations Example**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
.............
SIGNAL a: IN SIGNED (7 DOWNTO 0);
SIGNAL b: IN SIGNED (7 DOWNTO 0);
SIGNAL x: OUT SIGNED (7 DOWNTO 0);
SIGNAL u: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL v: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL y: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
.............
x <= a + b;                    - -legal
x = a AND b;                   - - illegal
y = a + b;                     - - illegal
y = a AND b;                   - - legal
```

VHDL Data Types:
Signed and Unsigned Types

- std_logic_signed and std_logic_unsigned packages allows both logical and arithmetic operations

- Example:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
...
SIGNAL a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL x: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
v <= a+ b;                    -- legal
w <= a AND b;                 -- legal
```

Type Conversion

- Direct operation between different data types is illegal in VHDL
- Solution!!!! = Data conversion
- Examples:

```
TYPE long IS INTEGER RANGE -100 TO 100;
TYPE short IS INTEGER RANGE -10 TO 10;
SIGNAL x : short;
SIGNAL y : long;
...
y <= 2*x + 5;          -- error, type mismatch
y <= long(2*x + 5);    -- OK, result converted into type long
```

Type Conversion

- Data conversion defined in STD_LOGIC_ARITH
- conv_integer(p) :
  - Converts a parameter p of type INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC to an INTEGER value.
  - Notice that STD_LOGIC_VECTOR is not included.

- conv_unsigned(p, b):
  Converts a parameter p of type INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC to an UNSIGNED value with size b bits.

- conv_signed(p, b):
  Converts a parameter p of type INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC to a SIGNED value with size b bits.

Type Conversion

- conv_std_logic_vector(p, b):
  - Converts a parameter p of type INTEGER, UN-SIGNED, SIGNED, or STD_LOGIC to a STD_LOGIC_VECTOR value with size b bits.
- Example:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
...
SIGNAL a: IN UNSIGNED (7 DOWNTO 0);
SIGNAL b: IN UNSIGNED (7 DOWNTO 0);
SIGNAL y: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
y <= CONV_STD_LOGIC_VECTOR ((a+b), 8);
```

- a+b is converted from UNSIGNED to an 8-bit STD_LOGIC_VECTOR value, then assigned to y.

VHDL Data Type:Example

- **Four Bit Adder** **------ Solution 2: in/out=SIGNED ----------**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
---------------------------------------
ENTITY adder1 IS
    PORT ( a, b : IN SIGNED (3 DOWNTO 0);
    sum : OUT SIGNED (4 DOWNTO 0));
END adder1;
---------------------------------------
ARCHITECTURE adder1 OF adder1 IS
    BEGIN
    sum <= a + b;
END adder1;
```

VHDL Data Type:Example

Four Bit Adder------ Solution 2: out=INTEGER ----------

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
---------------------------------------
ENTITY adder2 IS
    PORT ( a, b : IN SIGNED (3 DOWNTO 0);
           sum : OUT INTEGER RANGE -16 TO 15);
END adder2;
---------------------------------------
ARCHITECTURE adder2 OF adder2 IS
    BEGIN
        sum <= CONV_INTEGER(a + b);
END adder2;
```