# Integrated Circuits Design by FPGA

م.م. أحمد مؤيد عبدالحسين
جامعة الفرات الأوسط التقنية / الكلية التقنية الهندسية / نجف

# Lecture 2

## Lecture 2 : VHDL Data Classes and Data Types

# Objectives of this Lecture

- To describe the four basic data classes
- To describe the fundamental data types.
- To indicate what are the synthesizable data types.

# Contents of this Lecture

- **VHDL Data Classes.**
- **VHDL Data Types.**
- **Arrays**

# VHDL Data Classes

- As in any other programming language VHDL contains different data **classes** and data **types**.
- In VHDL there are 4 different Data classes

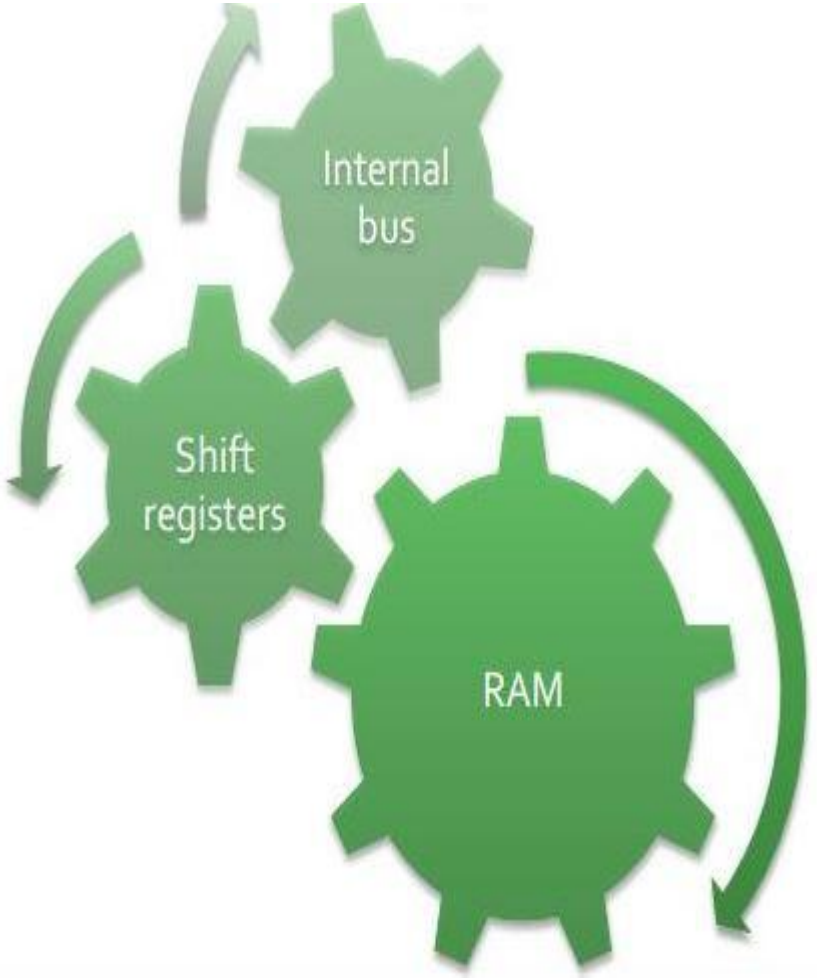
# VHDL Data Classes

## 1. Signals

- Signals are used to implement **internal buses**, **shift registers**, and **RAM.**

- Signals are assigned a value. The projected values can be changed as many times as desired.

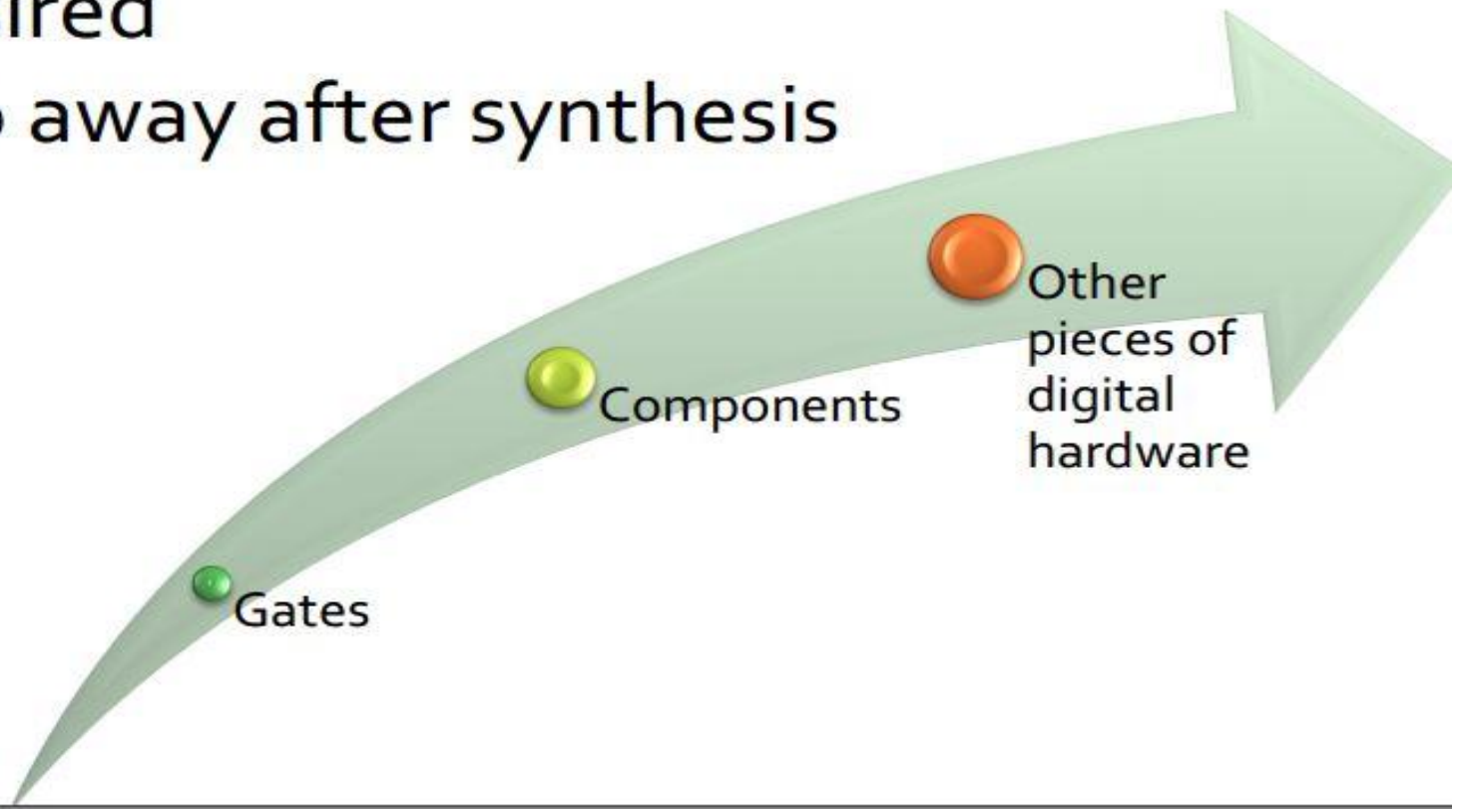
- Signals are synthesized in our circuit design.



# VHDL Data Classes

## 2. Variables

- An object with only a current value
- Variable values can be changed as many times as desired
- Variables go away after synthesis

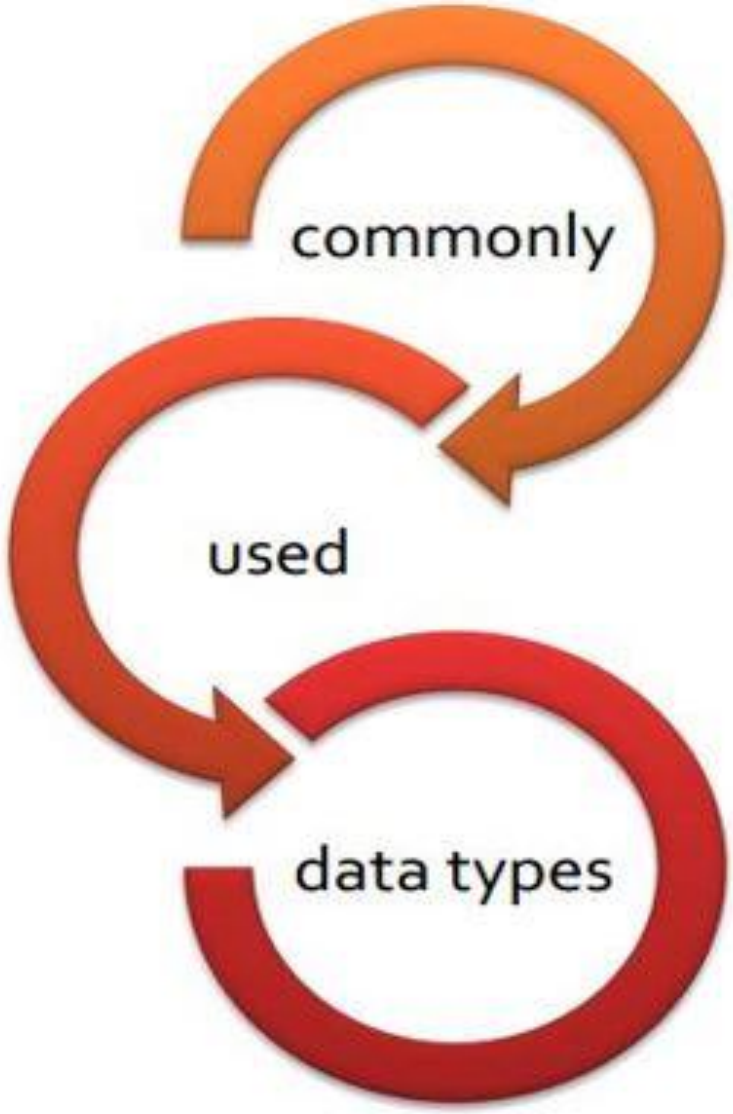


# VHDL Data Classes

**3. Constants :** As the name implies.

**4. Files:** An object that consists of a sequence of values. This object is used in simulation files only (Test Benches).

# VHDL Data Types




- std_logic
- std_logic_vector
- signed
- unsigned
- integer
- User defined

# VHDL Data Types : std_logic

- A “std_logic” type is either a logic “1” or a logic “0”.
- Defined in package STD_LOGIC_1164

```
signal shift                    : std_logic;
```

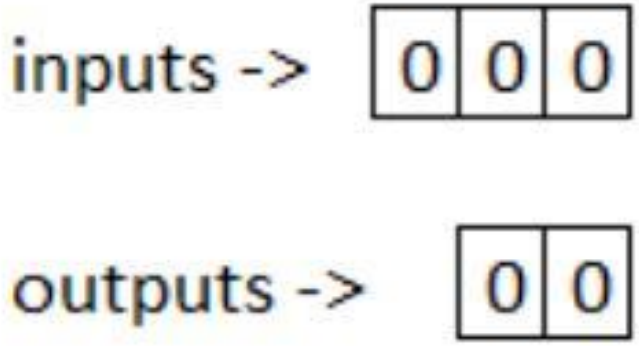
A standard logic data type is a single binary bit

shift -> 0

# VHDL Data Types : std_logic_vector

- Defined in package STD_LOGIC_1164
- A "std_logic_vector" is a binary representation of a number.
- For example:

```
signal inputs   : std_logic_vector(2 downto 0);
signal outputs  : std_logic_vector(1 downto 0);
```



# VHDL Data Types : signed

- Declared in package NUMERIC_STD
- Is an array of std_logic
- Type signed is interpreted as a signed binary number in twos complement form
- The left most (MSB) is the signed bit (1 = negative, 0 = positive)

```
-- Set signal A to -5
signal A :signed(3 downto 0) := '1101';

-- Set signal B to +5
signal B :signed(3 downto 0) := '0101';
```

# VHDL Data Types : unsigned

- Declared in package NUMERIC_STD
- Is an array of std_logic
- Type signed is interpreted as a std_logic_vector but has different operators available
- Can only represent positive numbers
- The left most bit is the MSB

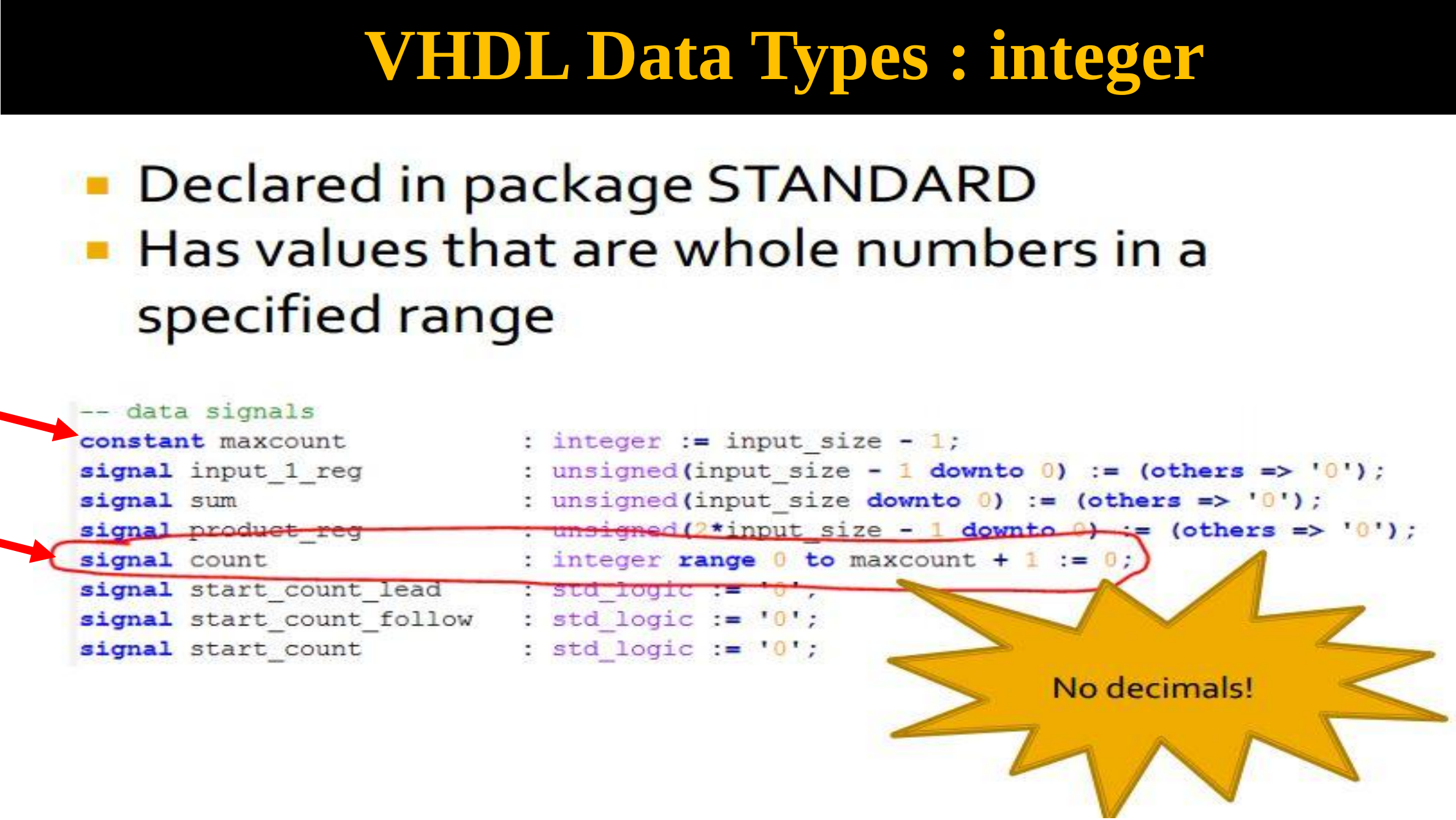
```
-- Set signal A to 13
signal A :unsigned(3 downto 0) := '1101';

-- Set signal B to 5
signal B :unsigned(3 downto 0) := '0101';
```

# VHDL Data Types : integer

- Declared in package STANDARD
- Has values that are whole numbers in a specified range

```
-- data signals
constant maxcount           : integer := input_size - 1;
signal input_1_reg          : unsigned(input_size - 1 downto 0) := (others => '0');
signal sum                  : unsigned(input_size downto 0) := (others => '0');
signal product_reg          : unsigned(2*input_size - 1 downto 0) := (others => '0');
signal count                : integer range 0 to maxcount + 1 := 0;
signal start_count_lead     : std_logic := '0';
signal start_count_follow   : std_logic := '0';
signal start_count          : std_logic := '0';
```



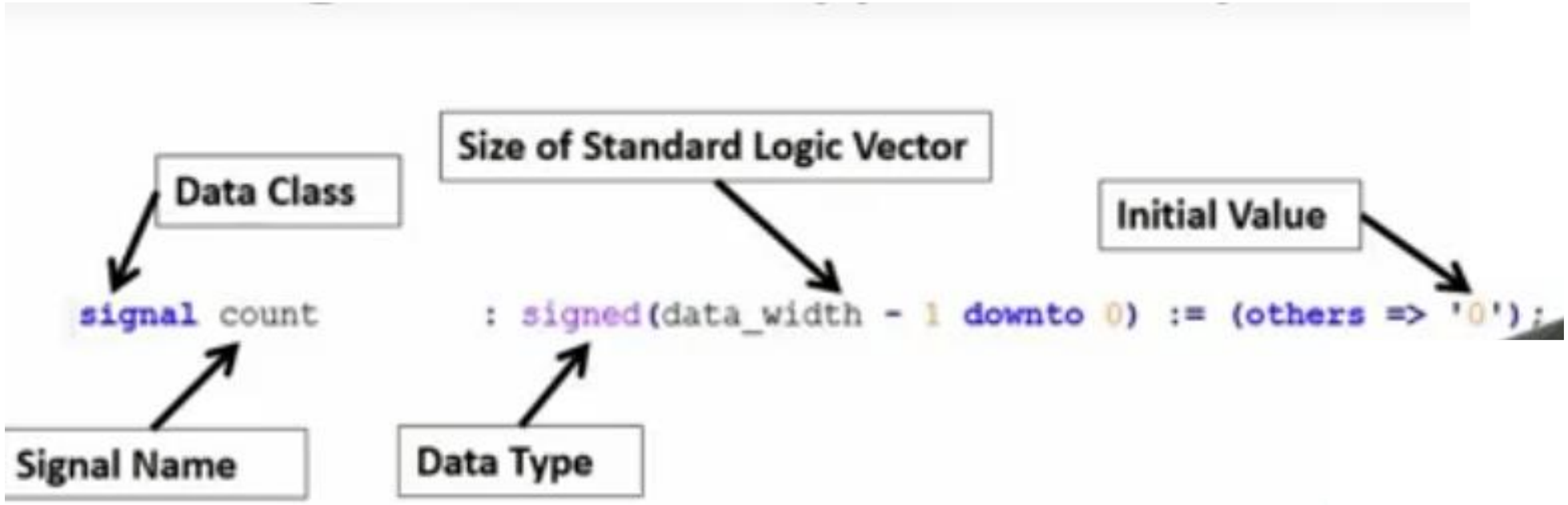
# VHDL Data Types : user defined

- Typically used in state machines

```
type state_type is(init, load, right_shift, done);
signal state, nxt_state : state_type;
```

# VHDL Code Structure

# VHDL Code Structure



# VHDL Code Structure : generic

- Allow a design entity to be altered by the choice of these values
- They are typically an integer or Boolean value

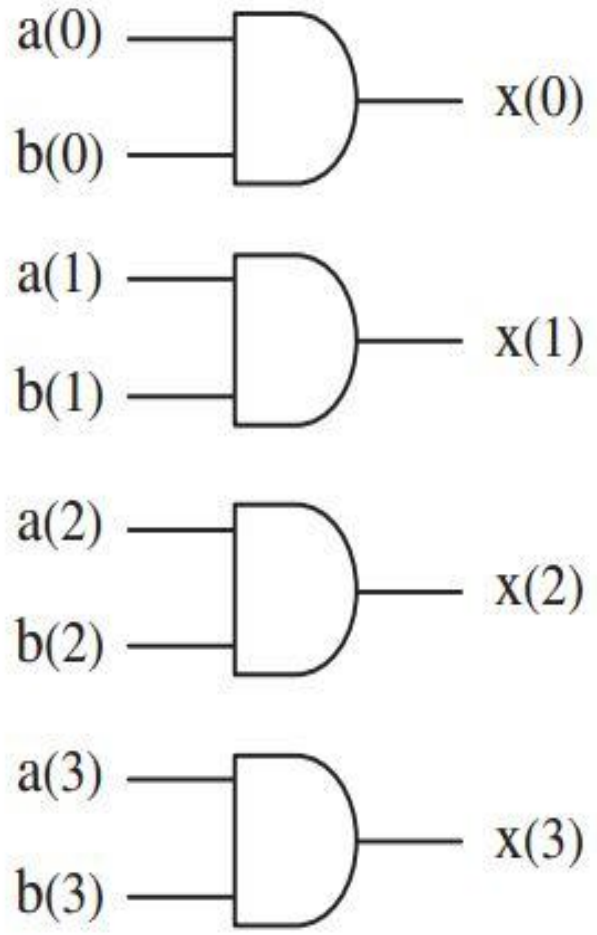
```
entity USR is
generic (
    data_width  : integer := 8);
port (
    A           : out std_logic_vector(data_width - 1 downto 0);
    I           : in std_logic_vector(data_width - 1 downto 0);
    S           : in std_logic_vector(2 downto 0);
    reset       : in std_logic;
    clk         : in std_logic);
end USR;
```

# Example 1

```
1  Library ieee;
2  use.ieee.std_logic_1164.all;
3  use.ieee.numeric_std.all;
4
5  ENTITY and2 IS
6  PORT (a, b: IN std_logic;
7  x: OUT std_logic);
8  END and2;
9  ------------------------------
10 ARCHITECTURE and2 OF and2 IS
11 BEGIN
12 x <= a AND b;
13 END and2;
```

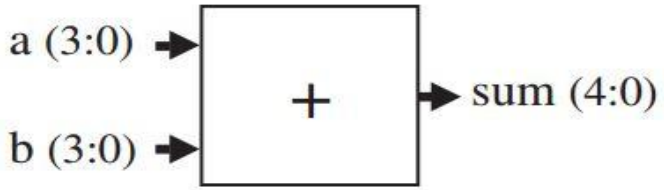
# Example 2

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
-------------------------------------
ENTITY and2 IS
PORT (a, b: IN std_logic_vector(3 downto 0);
x: OUT std_logic_vector(3 downto 0));
END and2;
-------------------------------------
ARCHITECTURE and2 OF and2 IS
BEGIN
x <= a and b;
END and2;
```



# Example 3

```
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE ieee.std_logic_arith.all;
5  ---------------------------------
6  ENTITY adder1 IS
7  PORT ( a, b : IN SIGNED (3 DOWNTO 0);
8  sum : OUT SIGNED (4 DOWNTO 0));
9  END adder1;
10 ---------------------------------
11 ARCHITECTURE adder1 OF adder1 IS
12 BEGIN
13 sum <= a + b;
14 END adder1;
```



# Arrays

- **Arrays are collections of objects of the same type.**

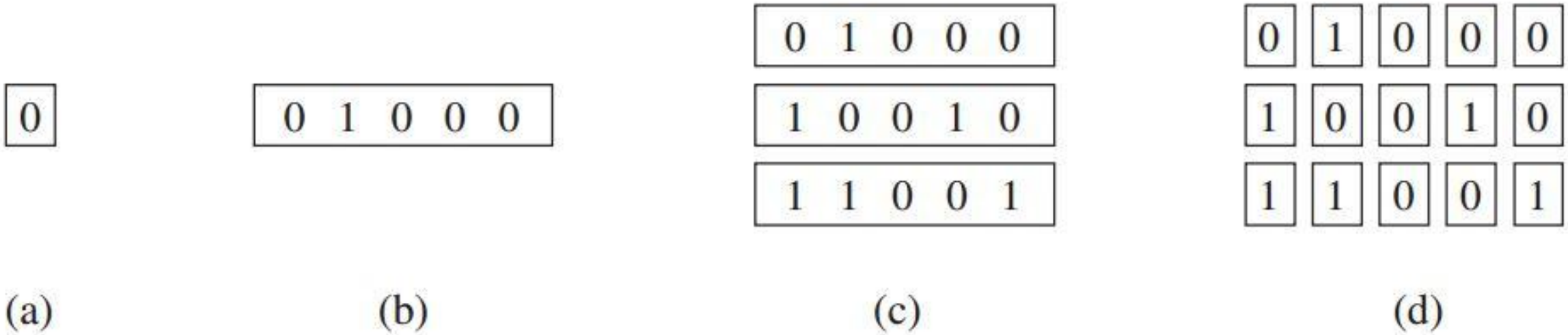



**Figure 3.1**
Illustration of (a) scalar, (b) 1D, (c) 1Dx1D, and (d) 2D data arrays.

# Arrays

```
signal a     :std_logic :='0';                          -- scalar
TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;            -- 1D array
TYPE matrix IS ARRAY (0 TO 3) OF row;                   -- 1Dx1D array
SIGNAL x: matrix;                                       -- 1Dx1D signal
TYPE matrix2D IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;   -- 2D array
```

# Assignments

- The assignments will be attached in your class room.

# End of lecture 2

# Any Questions ?