Module 2

VHDL Data Types

Data Types

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

Signals

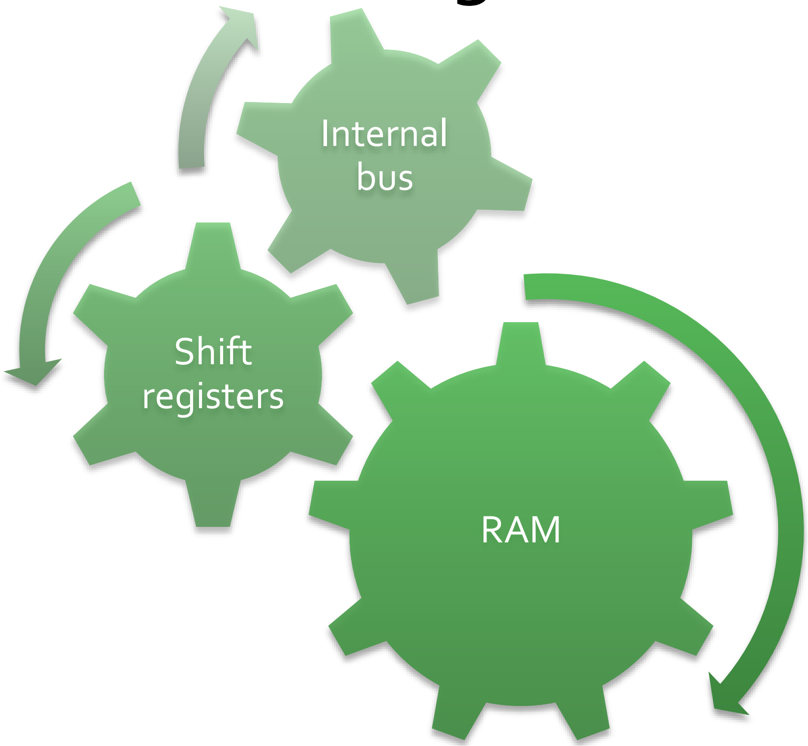
Constants

Variables

Files

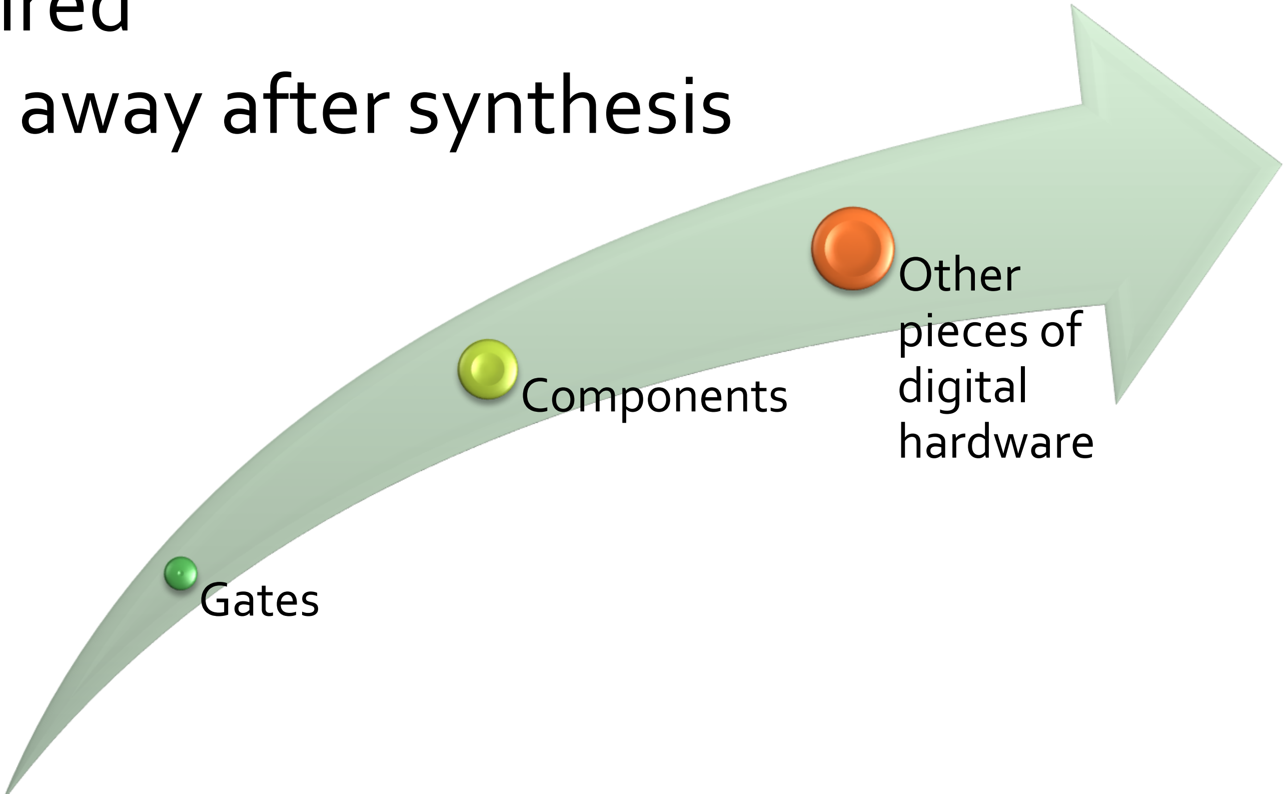
Signals

- An object with the current value and projected values
- The projected values can be changed as many times as desired



Variables

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



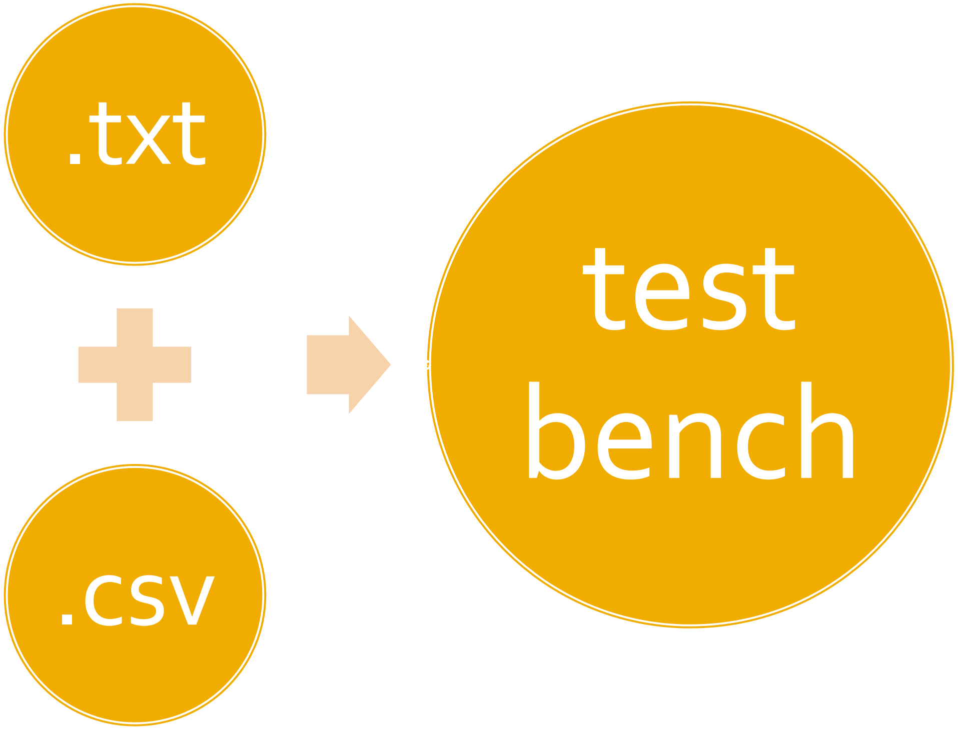
Gates

Components

Other pieces of digital hardware

Files

- An object that consists of a sequence of values
- These are only used in test benches



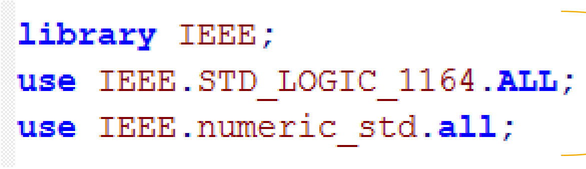
.txt

.csv

test bench

Libraries

- Library IEEE;
  - IEEE.std_logic_1164.all;
    - Allows the use of type “std_logic”
    - Allows the use of type “std_logic_vector”
  - IEEE.numeric_std.all;
    - Allows the use of type ”unsigned”
    - Allows the use of type “signed”

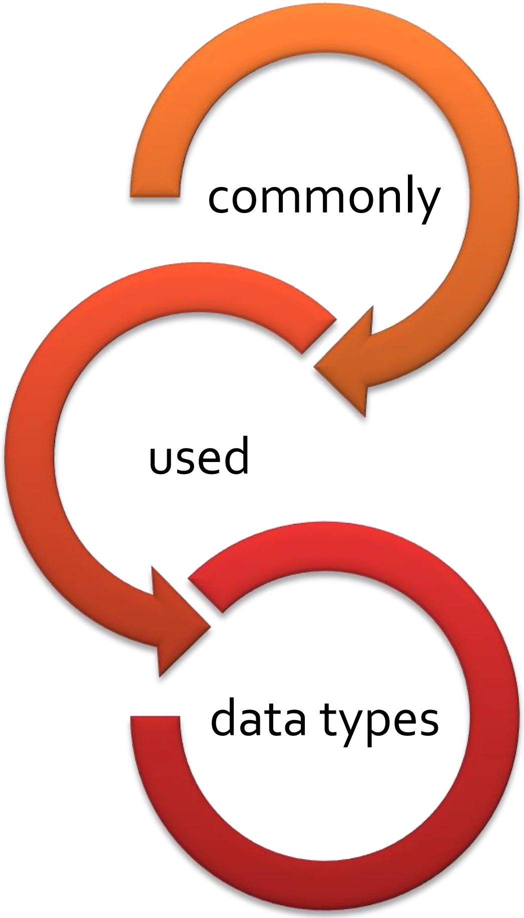


```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
```

Example of VHDL code

Different Data Types

- std_logic_vector
- std_logic
- signed
- unsigned
- integer
- boolean
- constant
- signal
- port
- generic
- user defined



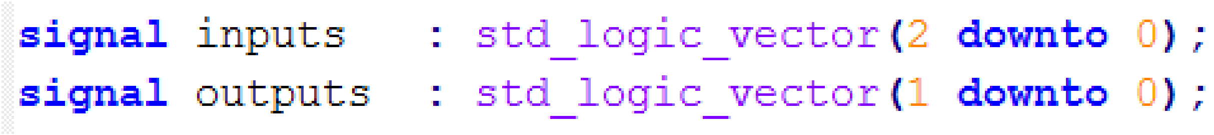
commonly

used

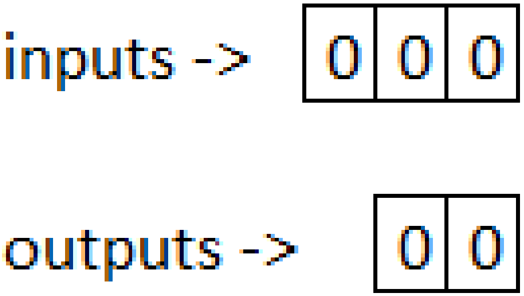
data types

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);
```

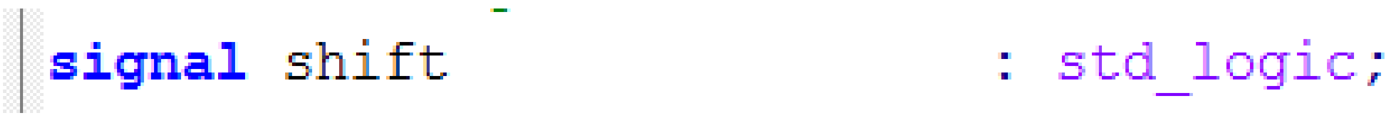


inputs -> 0 0 0

outputs -> 0 0

std_logic

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



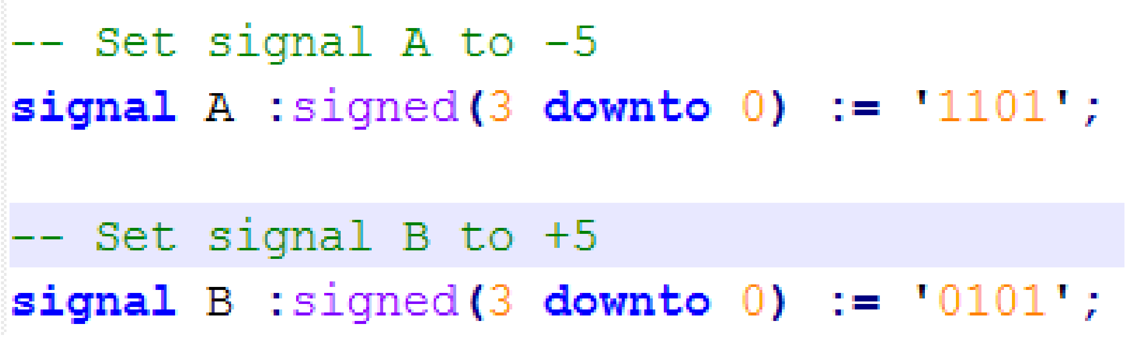
```
signal shift                    : std_logic;
```

A standard logic data type is a single binary bit

shift -> $\boxed{0}$

signed

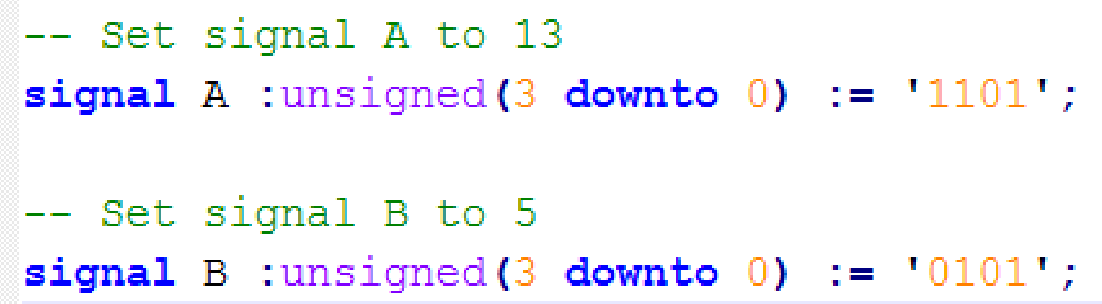
- Declared in package NUMERIC_STD
- Is an array of std_logic
- Type signed is interpreted as a signed binary number in twos complement form
- The left most (MSB) is the signed bit (1 = negative, 0 = positive)



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

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

unsigned

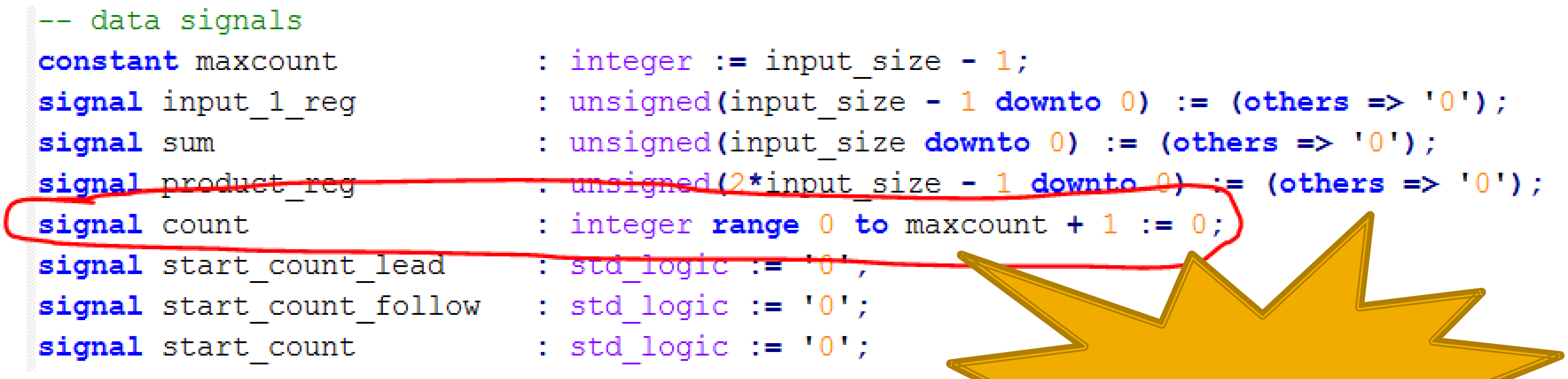
- Declared in package NUMERIC_STD
- Is an array of std_logic
- Type signed is interpreted as a std_logic_vector but has different operators available
- Can only represent positive numbers
- The left most bit is the MSB



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

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

integer

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

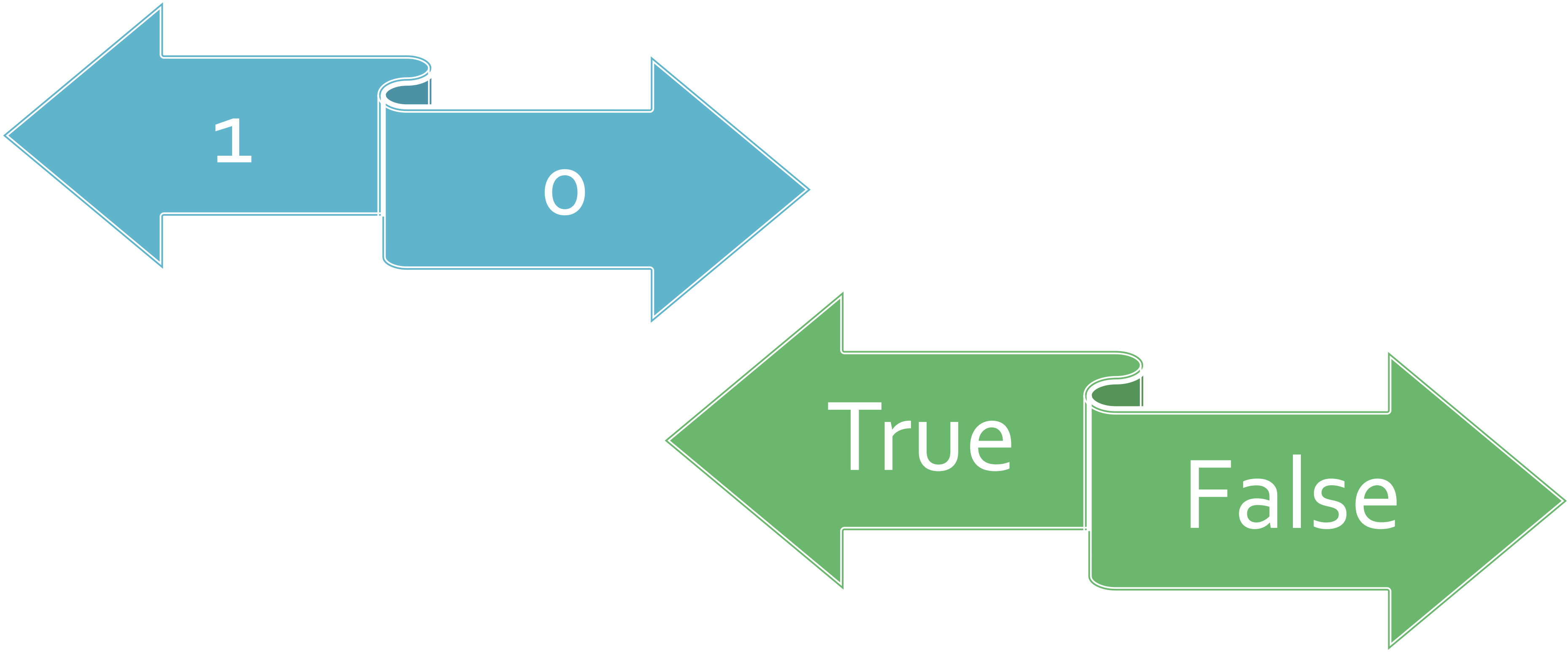


```
-- data signals
constant maxcount            : integer := input_size - 1;
signal input_1_reg           : unsigned(input_size - 1 downto 0) := (others => '0');
signal sum                   : unsigned(input_size downto 0) := (others => '0');
signal product_reg           : unsigned(2*input_size - 1 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';
```

No decimals!

boolean

- Declared in package STANDARD
- Can be either a 1 or 0, true or false



1

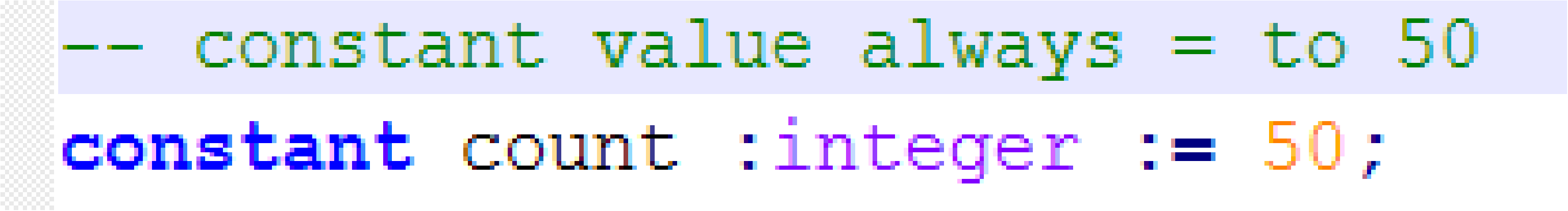
0

True

False

constant

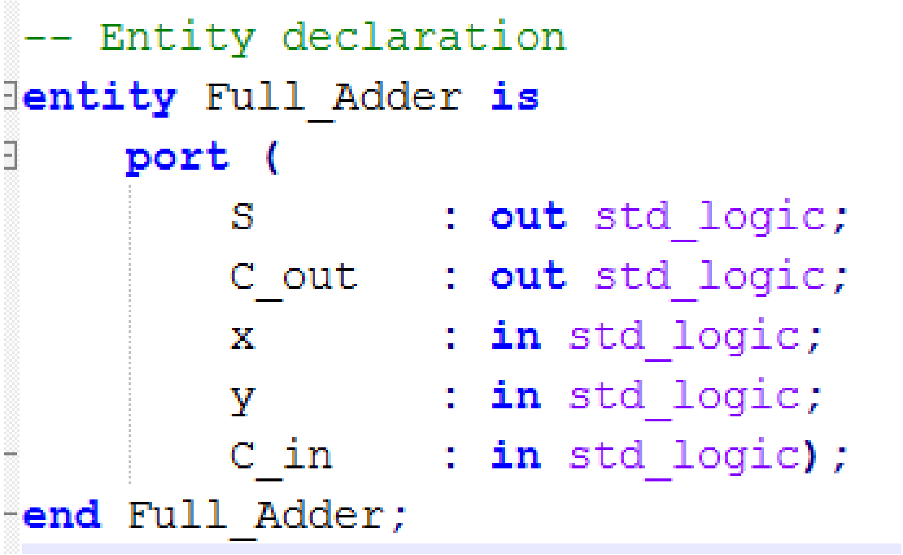
- An object whose value cannot be changed after it is initially specified



```
-- constant value always = to 50
constant count :integer := 50;
```

port

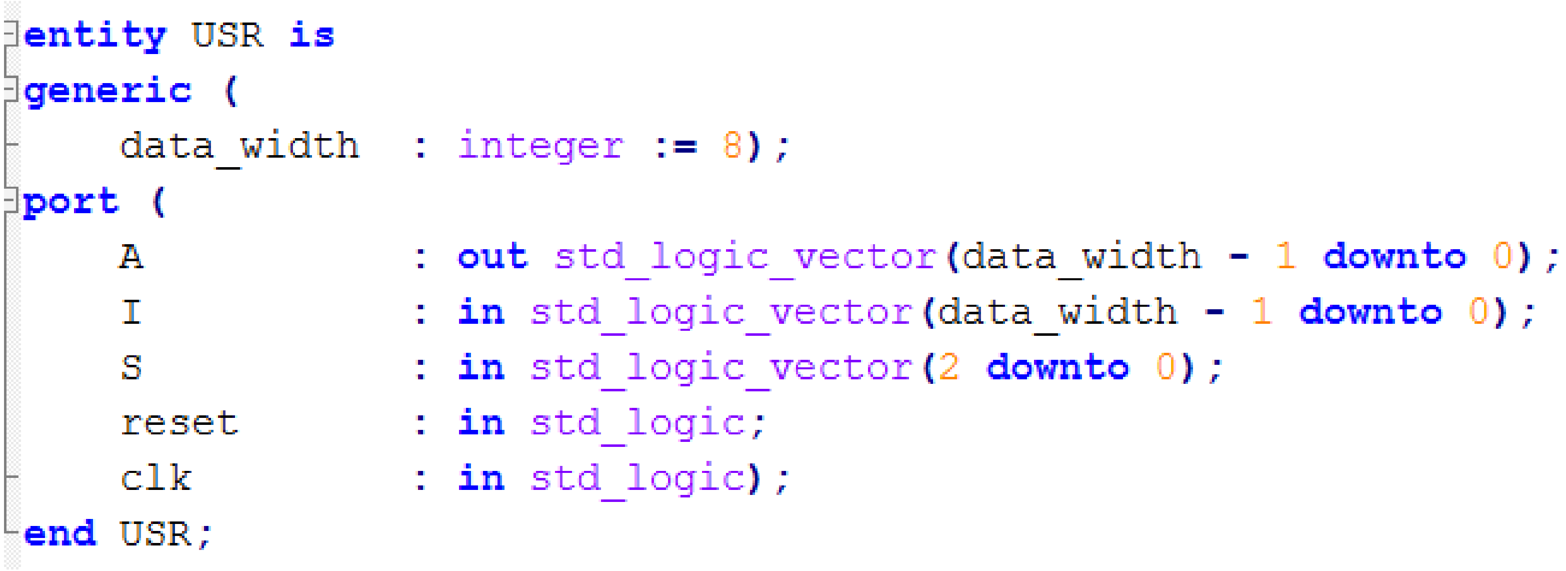
- Used to define your input and output user interface signals
- Located in the entity section of your VHDL design



```
-- Entity declaration
entity Full_Adder is
    port (
        S       : out std_logic;
        C_out   : out std_logic;
        x       : in std_logic;
        y       : in std_logic;
        C_in    : in std_logic);
end Full_Adder;
```

generic

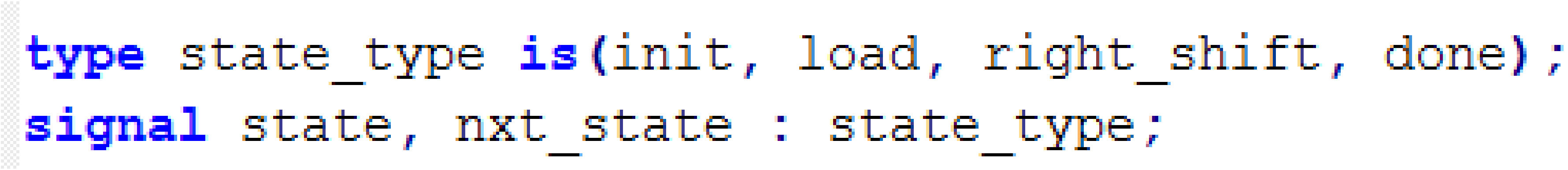
- Allow a design entity to be altered by the choice of these values
- They are typically an integer or Boolean value



```
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;
```

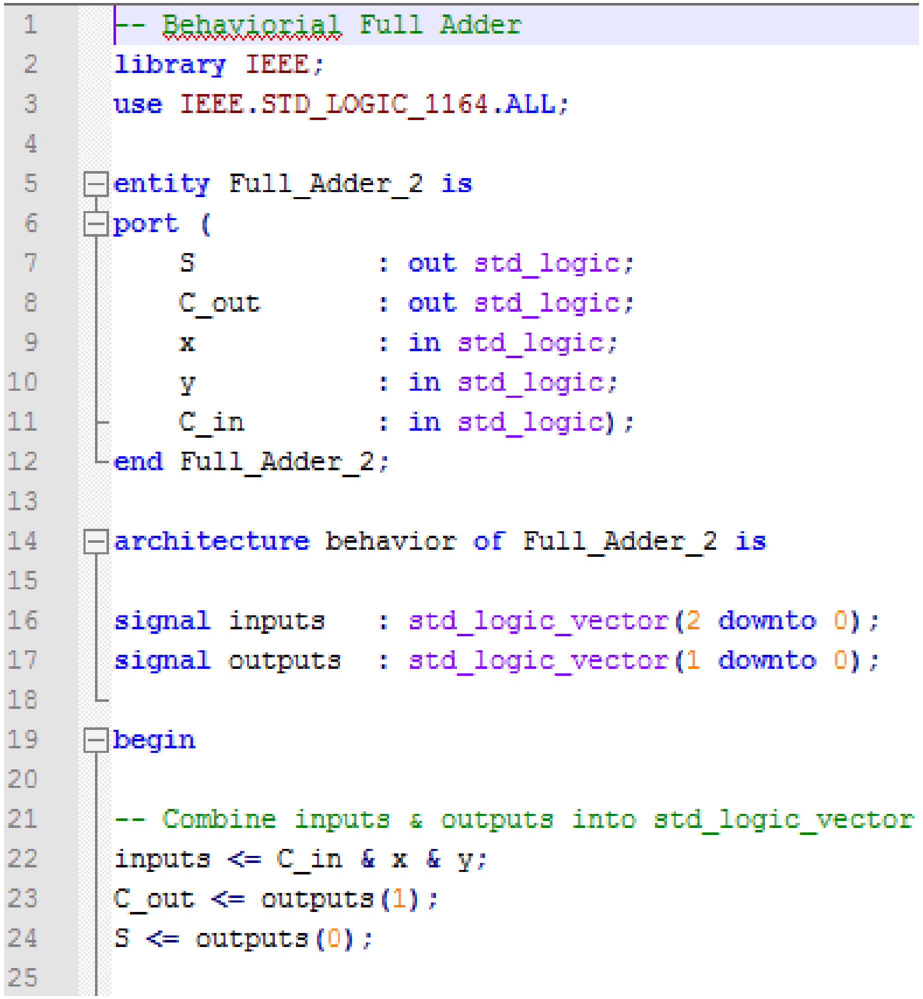
User defined

- Typically used in state machines

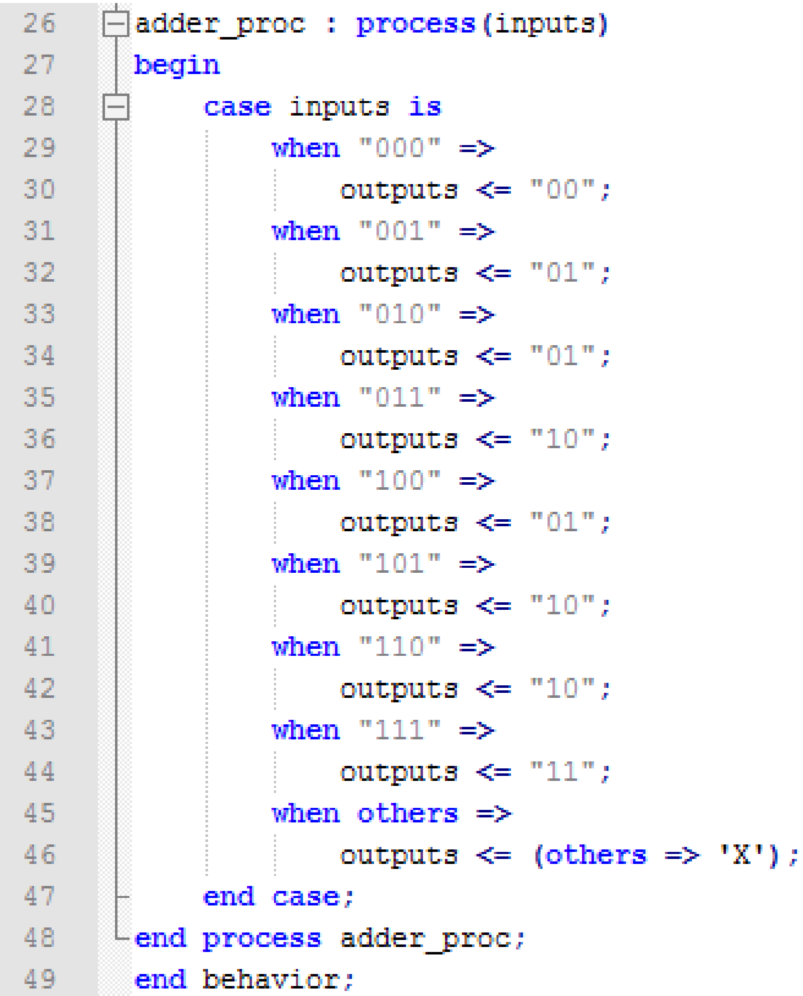


```
type state_type is(init, load, right_shift, done);
signal state, nxt_state : state_type;
```

Example



```vhdl
-- Behaviorial Full Adder
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Full_Adder_2 is
port (
    S           : out std_logic;
    C_out       : out std_logic;
    x           : in std_logic;
    y           : in std_logic;
    C_in        : in std_logic);
end Full_Adder_2;

architecture behavior of Full_Adder_2 is

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

begin

-- Combine inputs & outputs into std_logic_vector
inputs <= C_in & x & y;
C_out <= outputs(1);
S <= outputs(0);

```



```vhdl
adder_proc : process(inputs)
begin
    case inputs is
        when "000" =>
            outputs <= "00";
        when "001" =>
            outputs <= "01";
        when "010" =>
            outputs <= "01";
        when "011" =>
            outputs <= "10";
        when "100" =>
            outputs <= "01";
        when "101" =>
            outputs <= "10";
        when "110" =>
            outputs <= "10";
        when "111" =>
            outputs <= "11";
        when others =>
            outputs <= (others => 'X');
    end case;
end process adder_proc;
end behavior;
```

Synthesis

- Synthesis is when your VHDL design is "synthesized" by the specific programmable logic devices tool to be placed onto the target FPGA / CPLD/ etc…
- Synthesis simply put is taking your VHDL design and formatting it so that the FPGA or CPLD knows which gates to place and where to place them.

Summary

Signed values are used to represent negative values.

Std_logic_vectors are binary values that allow you to manipulate bits individually.

Items placed in the Generic section will not show up as in input or output.

Types need to match when performing operations.

Std_logic_vector + Unsigned is not allowed.