FACULTY OF ENGINEERING

CAIRO UNIVERSITY

LAB 4
INTRODUCTION TO VHDL

CMP301A: COMPUTER ARCHITECTURE COURSE

EXERTED FROM DR. ADNAN SHAOUT- *THE UNIVERSITY OF MICHIGAN-DEARBORN*

OBJECTIVES

- Declaring New Types.
- How to initialize Memory.
- Test-benches.

MEMORY / NEW TYPES …

- Let’s Define the entity of the Ram Block of size 64 byte
- What are the (input/outputs) for this circuit?
- How many bits should represent each Input/Output ?

MEMORY / NEW TYPES …

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.numeric_std.all;
ENTITY ram IS
 PORT (clk : IN std_logic;
       we : IN std_logic;
    address : IN std_logic_vector(5 DOWNTO 0);
    datain  : IN std_logic_vector(7 DOWNTO 0);
    dataout : OUT std_logic_vector(7 DOWNTO 0) );
END ENTITY ram;
```

What is this library ?!

Write/read signal

```
ARCHITECTURE sync_ram_a OF ram IS
  TYPE ram_type IS ARRAY(0 TO 63) of std_logic_vector(7 DOWNTO 0);
     SIGNAL ram : ram_type ;
BEGIN
PROCESS(clk) IS
BEGIN
    IF rising_edge(clk) THEN
         IF we = '1' THEN
         ram(to_integer(unsigned((address))) <= datain;
         END IF;
   END IF;
END PROCESS;
       dataout <= ram(to_integer(unsigned((address)));
END sync_ram_a;
```

Defining a new type

Two Dimensional Array

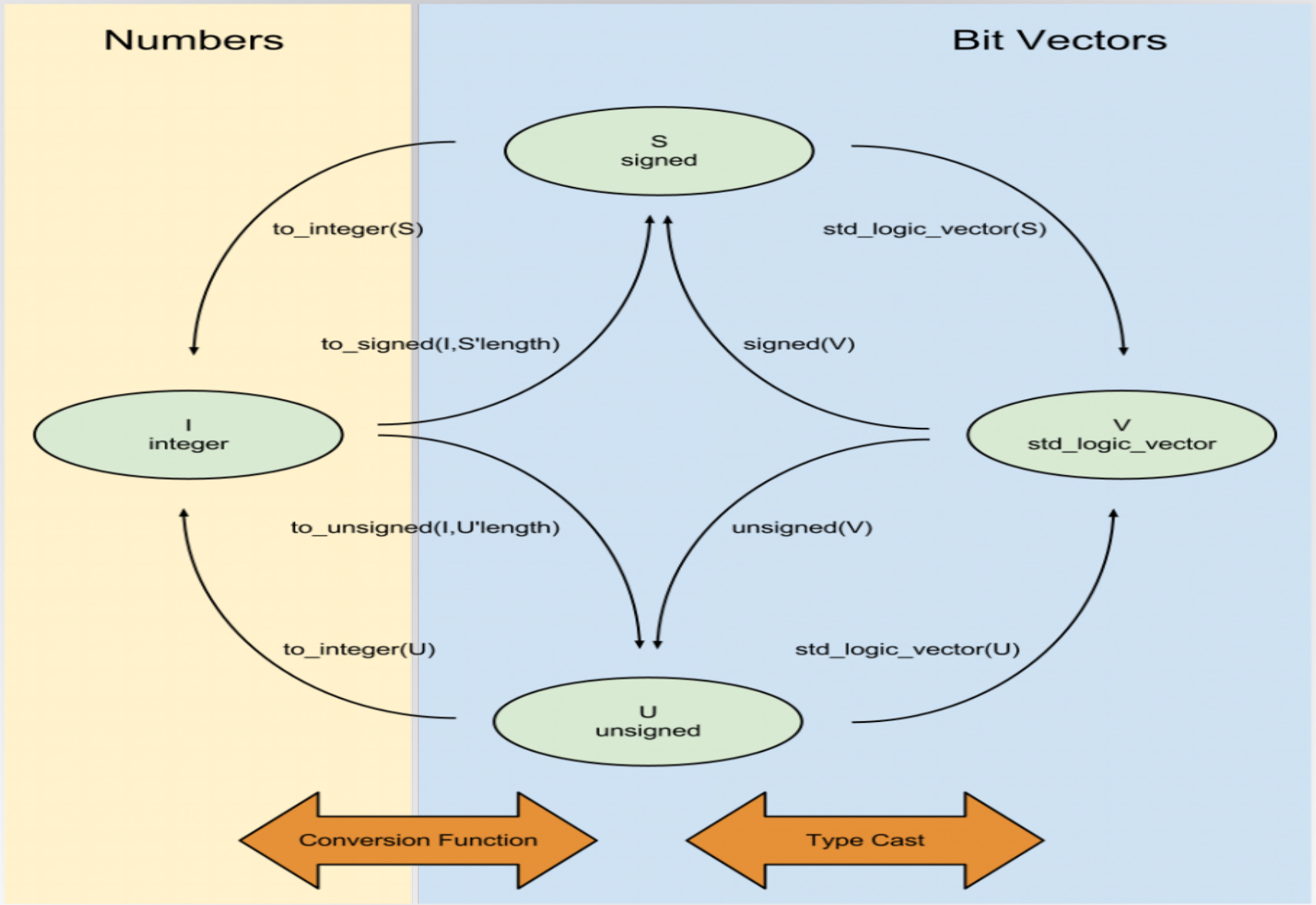
```
ARCHITECTURE sync_ram_a OF ram IS
  TYPE ram_type IS ARRAY(0 TO 63) of std_logic_vector(7 DOWNTO 0);
    SIGNAL ram : ram_type ;
BEGIN
PROCESS(clk) IS
BEGIN
   IF rising_edge(clk) THEN
       IF we = '1' THEN
       ram(to_integer(unsigned((address))) <= datain;
       END IF;
  END IF;
END PROCESS;
      dataout <= ram(to_integer(unsigned((address)));
END sync_ram_a;
```

Using the new Type

Remember the numeric Library ?! It contains this functions

What does synchronous mean ?!

Output here is asynchronous



HTTP://WWW.BITWEENIE.COM/LISTINGS/VHDL-TYPE-CONVERSION/

MODELSIM TIME !

TEST BENCH EXAMPLE

```
-- Component under test
ENTITY and2 IS
      PORT( a,b : IN std_logic;
                  z : OUT std_logic);
END and2;
```

```
ARCHITECTURE anda OF and2 IS
BEGIN
      z <= a AND b;
END and_a;
```

```
ENTITY testbench IS
END testbench;

ARCHITECTURE testbench_a OF testbench IS
    COMPONENT and2 IS
    PORT ( a,b : IN std_logic; z : OUT std_logic);
    END COMPONENT;
SIGNAL testa, testb, testz : std_logic;
BEGIN
PROCESS
BEGIN
    testa <= '0';
    testb <= '0';
    WAIT FOR 10 ns;
    ASSERT(testz = '0') REPORT  "z is not 0 for 00"
    SEVERITY ERROR;
    testa <= '0';
    testb <= '1';
    WAIT FOR 10 ns;
    ASSERT(testz = '0') REPORT "z is not 0 for 01"
    SEVERITY ERROR;
    testa <= '1';
    testb <= '0';
    WAIT FOR 10 ns;
    ASSERT(testz = '0') REPORT "z is not 0 for 10"
    SEVERITY ERROR;
    testa <= '1';
    testb <= '1';
    WAIT FOR 10 ns;
    ASSERT(testz = '1') REPORT "z is not 1 for 11"
    SEVERITY ERROR;
WAIT;
END PROCESS;

uut: and2 PORT MAP (a => testa, b => testb,   z =>testz);

END testbench_a;
```

```
ENTITY testbench IS
END testbench;

ARCHITECTURE testbench_a OF te
    COMPONENT and2 IS
    PORT ( a,b : IN std_logic; z : OUT std_logic);
    END COMPONENT;
SIGNAL testa, testb, testz : std  logic;
BEGIN
PROCESS
BEGIN
    testa <= '0';
    testb <= '0';
    WAIT FOR 10 ns;
    ASSERT(testz = '0') REPORT  "z is not 0 for 00"
    SEVERITY ERROR;
    testa <= '0';
    testb <= '1';
    WAIT FOR 10 ns;
    ASSERT(testz = '0') REPORT "z is not 0 for 01"
    SEVERITY ERROR;
    testa <= '1';
    testb <=
    WAIT
    ASSERT                                  "z is  t 0 for 10"
    SEVERITY ER     R;
                ';
    testb <= '1';
    WAIT FOR 10 ns;
    ASSERT(testz = '1') REPORT "z is not 1 for 11"
    SEVERITY ERROR;
WAIT;
END PROCESS;

uut: and2 PORT MAP (a => testa, b =>
testb,   z =>testz);

END testbench_a;
```

Empty entity

Entity under test

Why used signals instead of variables ?!

Variables will not be seen outside process

Process with No sensitivity list ?!

Remember Signal takes time to get value

If the condition is true , then it doesn't report

To stop looping forever

Instantiating the component to be tested

The process and the component works in parallel

Can’t We just use a loop instead of writing cases ?

```
PROCESS
    CONSTANT outCases : std_logic_vector(0 to 3) :="0001";
BEGIN
    For i in 0 to 3 Loop
        test(2 downto 1) <= std_logic_vector(to_unsigned(i,2));
        WAIT FOR 10 ns;
        ASSERT(test(0) = outCases(i))      REPORT  " Z is not "
                    &std_logic'image(outCases(i))&" zero for "&integer'image(i)
        SEVERITY ERROR;
    end loop;
    WAIT;
END PROCESS;
```