Hardware Design using VHDL
Lab 3

1. Introduction

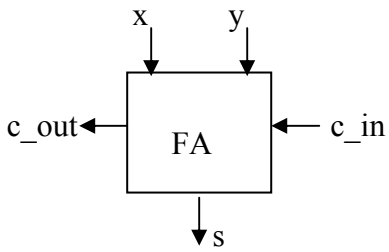
This laboratory consists of two parts. In **part 1,** the use **of structural VHDL.** The design is simulated and implemented in a gate array. **Part 2** consists of **simulation of** a **matrix-vector multiplication.**

2. Preparation

Read Sjoholm-Lindh chapter 6 and 7. Read the lab-PM.
Perform design work for problem 1-2.

3. Details

Problem 1: Structural VHDL



a) Design a component FA (Full Adder), and compile.
$x + y + c\_in = 2 \cdot c\_out + s$

b) Design a 3-bits full adder using the component FA, using **structural VHDL.**

c) Simulate. Write the test bench carefully!

d) Synthesize. Note the maximum **delay time** and **area consumption** (number of LUT:s).

e) Implement. DIP1-3 to $X = \langle x_2, x_1, x_0 \rangle$ and DIP6-8 to $Y = \langle y_2, y_1, y_0 \rangle$.
c_in to DIP4.

The result c_out and $S = \langle s_2, s_1, s_0 \rangle$ is detected in decimal form on the 7-segment displays.

Problem 2: Matrix-vector multiplication in a systolic array

Introduction

A systolic array is a type of parallel architecture in which data is "pumped" through a series of processors linked by a field. The word "systolic" is the same as those used in medicine to enter the (systolic) blood pressure. A global clock set the pace, i.e. when data is shifted from one processor to the next. All processors perform the same operation.
In this problem a matrix-vector multiplication on a systolic array is going to be studied by simulation in VHDL.

Preparation before lab-moment

Make a solution "on paper" **before you** come to the lab.

Matrix-vector multiplication

Mathematically, we can take out the matrix-vector multiplication as:

$$C_k = \Sigma A_{ki} \times B_i$$

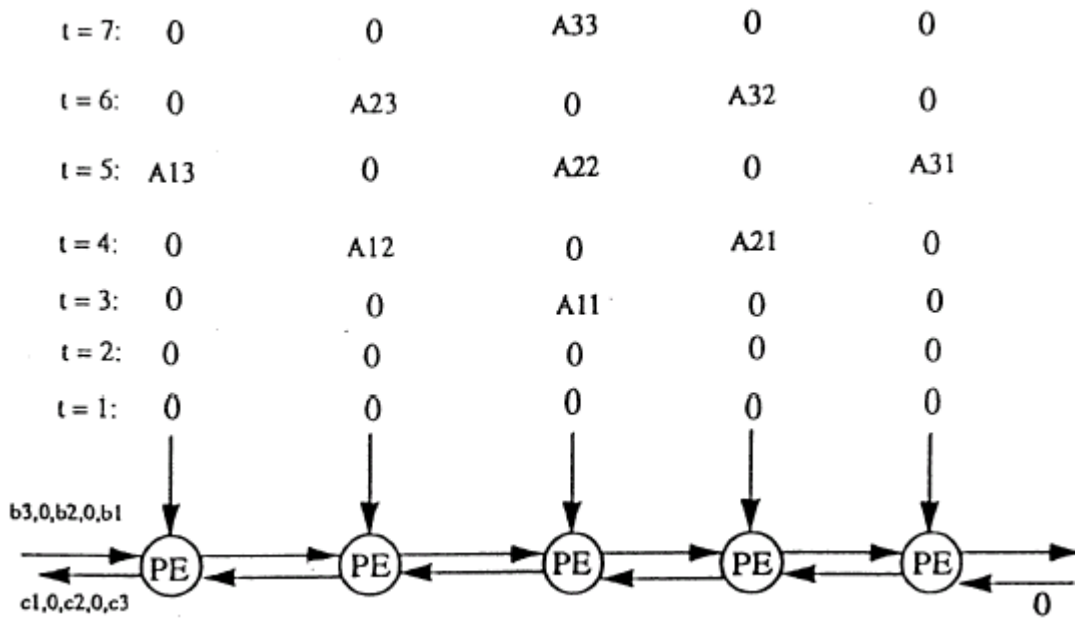
Here, A is a n x n matrix, B and C are vectors of dimension n x 1.

If for example we shall multiply a 3 x 3 matrix with a 3 x 1 vector, we can give this as:

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Systolic array

The systolic arrays in this case consists of five processor elements. Each element performs the same operation(s). These are:



1) Input from the left is copied out to the right (Right_out <= Left_in).

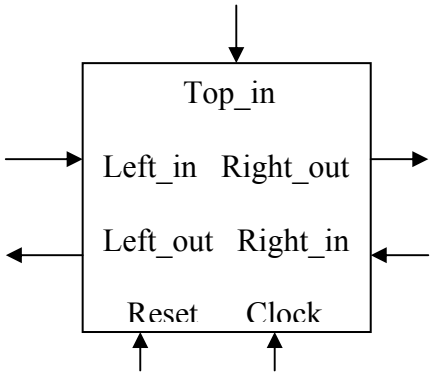
2) Output to the left is calculated as: Left_out <= (Left_in * Top_in) + Right_in.

The B-vector is pumped in from the left in the sequence: b1, 0, b2, 0, b3. The A-matrix is pumped into from above as the sequence shown in the figure. Utdatavektorn C comes out to left under the sequence c1, 0, c2, 0, c3.

Let the output of each processor be delayed eg 7 ns after clock flank. Clock the model with 10 ns periodtid (100 MHz).

VHDL modeling

A processor element is described in figure below. All signals, **except for clock and reset,** must be of type *integer.*



The complete array of five PE, is shown in the figure on the previous page.

Execution

This lab is based on knowledge's and methods that was trained in the previous labs. It may therefore be good to look at lab1 Dice.

Start the symbol editor and create symbol PE as shown in Fig. All signals, except **clock** and **reset,** must be of type *integer.*

When the PE component is created, it shall be tested. Start the schematic editor and place **one** PE. Create the I/O ports required and then save the schematic with the name **onepe.** Create a symbol of this schematic, and then quit the schematic editor.

Create the test bench and edit it.Add the appropriate VHDL code that tests the component PE

Compile and run the simulator. Check that the component PE works as it was thought. Correct any errors.

When the PE works the full matrix-vector multiplier is going to be created. Create a new schematic with an appropriate name (i.e. systolic) with five PE connected as shown in fig. Create a new test bench to test the whole array and simulate this.

Test the component with the following values:

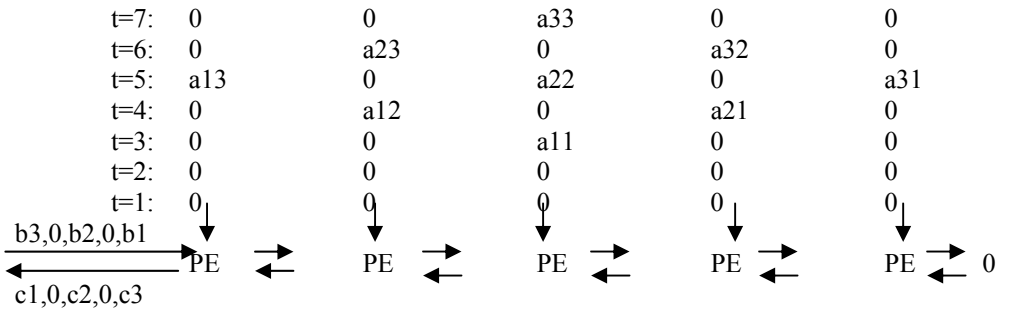
$$\begin{bmatrix} 3 & 4 & 5 \\ 1 & 2 & 3 \\ 0 & 6 & 7 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 26 \\ 14 \\ 33 \end{bmatrix}$$

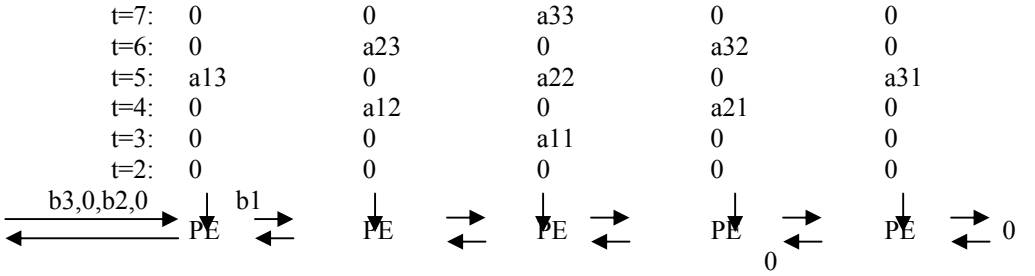
Accounting

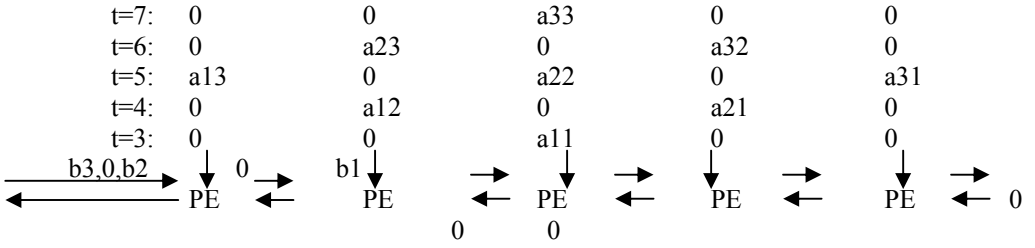
Plot the schematic and wave form.

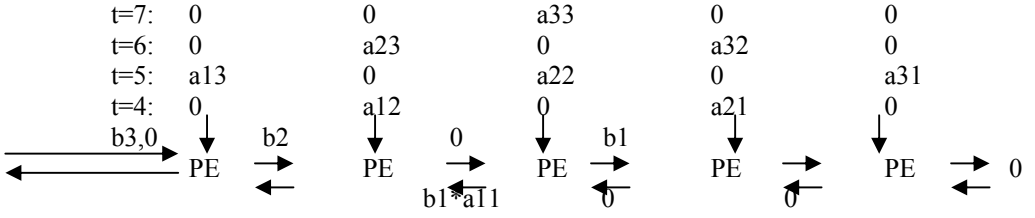
Appendix 1.

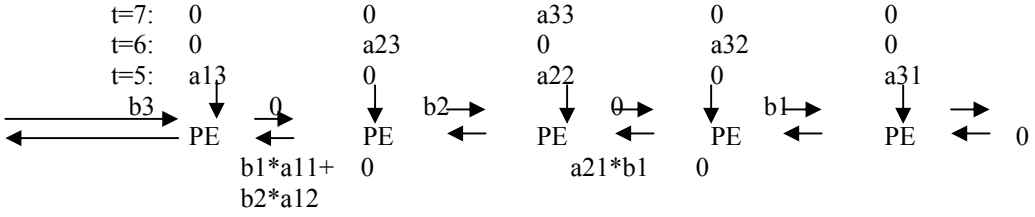
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_{11} b_1 & + a_{12} b_2 + a_{13} b_3 \\ a_{21} b_1 & + a_{22} b_2 + a_{23} b_3 \\ a_{31} b_1 & + a_{32} b_2 + a_{33} b_3 \end{bmatrix}$$

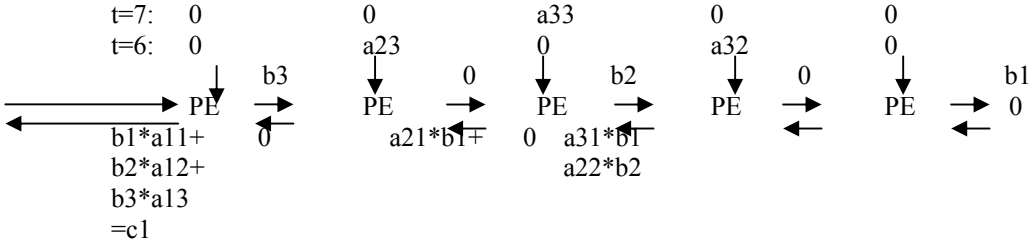












```
Library                 IEEE;

Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_unsigned.all;

constant numb_addr: integer := 7;
type pe_data is array (0 to numb_addr - 1, 0 to 4) of integer;

constant pe_matrix:
            pe_data:=pe_data' ( 6=> (0, 0, 7, 0, 0),
                                                    5=> (0, 3, 0, 6, 0),
                                                    4=> (5, 0, 2, 0, 0),
                                                    3=> (0, 4, 0, 1, 0),
                                                    2=> (0, 0, 3, 0, 0),
                                                    1=> (0, 0, 0, 0, 0),
                                                    0=> (0, 0, 0, 0, 0);

type vec is array (0 to numb_addr-1) of integer;

constant pe_vec: vec := vec' (1, 0, 2, 0, 3, 0, 0);
```