**Further DIgital Electronics Project Report**

# Script Questions

## **Question 1:** What is the maximum width (in terms of bits) of the coefficients of C, as a function of the generic values in your system? Which generics are relevant?

* The value of coefficient C is determined by the formula :
* The width of the coefficient C is determined by the maximum value that C can take. The relevant generics are:
  + **Data\_size :** the Data\_size of the coefficients in matrix A and B determine the maximum/minimum value that these coefficients can take, and therefore also the maximum/minimum value of the product . This in turn influences the width of C. The larger the modulus of the larger the width of C.
  + **M :**  the product is summed M times in the MACC, each summation potentially further increasing/decreasing the value, and therefore potentially further increasing the modulus of C.
* The maximum width of C is given by : . Derivation:
  + **:** the matrix coefficients can be both positive and negative, this means that signed values need to be used. If we have a 4-bit signed value, the largest absolute value it can take is if the MSB is 1 ( 1000 = -8 , abs(-8) = 8). The number of bits of the coefficients in matrix A and B is given by data\_size. gives us the maximum absolute value that a single coefficient can have.
  + **:** The maximum absolute value of a single coefficient is then squared since the the A and B coefficients are multiplied and both the A coefficient and the B coefficient could have the maximum absolute value. The maximum absolute value that can be obtained is when two coefficients with the maximum negative number are multiplied (e.g -8 \* -8 = 64 )
  + **:** The resulting value is then multiplied M (number of coefficients in a row of A / a column of B) times since all the coefficients in a row of A and a column of B could have the maximum negative value.
  + The ceiling of the log base 2 of this gives us the number of bits needed to express the maximum value that can be obtained as an unsigned.
  + **:** 1 is added since an extra bit is needed to express the value as a signed ( bit needed for 2’s complement).

## **QUestion 2:** What is the size (width and depth) of the memories that store A, B, and C, as a function of M, N, H, and the size of the data? What is then the width of the address bus for each of the memories?

The width of the address Busses is given by size(width\*height-1). Width \* height gives the number of fields from 1 to max, but the first address is at 0, so 1 has to be subtracted in order to obtain a bus of the correct size.

### Matrix\_A\_ROM:

* **Width:** Data\_Size
* **Depth:** H\*M
* **Adress width:** size(H\*M-1) note: size() function from DigEng.vhd package

### Matrix\_B\_ROM:

* **Width:** Data\_Size
* **Depth:** N\*M
* **Adress width:** size(N\*M-1) note: size() function from DigEng.vhd package

### MATRIX\_C\_RAM:

* **Width:**
* **Depth:** N\*H
* **Adress width:** size(N\*M-1) note: size() function from DigEng.vhd package

## **Question 3:** **A)** How are the counters used for address generation related to each other? **B)** When should each counter reset to 0 (beyond the global reset)? **C)** How are the addresses computed from the counter values?

### HOW ARE THE COUNTERS USED FOR ADDRESS GENERATION RELATED TO EACH OTHER?

* The M counter indicates the current position in the
  + Column of matrix A
  + Row of matrix B
* The M counter will go through one entire count cycle for each coefficient in matrix C.
* The N counter indicates the current position in the columns of matrix B and the columns of matrix C.
* The N counter is enabled when the M counter reaches its maximum value.
* The H counter indicates the the row of matrix A and the row of matrix C.
* The H counter is enabled when the N counter reaches its maximum value.

### When should each counter reset to 0 (beyond the global reset)?

* The M counter resets back to 0 once it reaches its maximum value. This happens every time the computation of a coefficient of the C matrix finishes.
* The N counter resets back to 0 once it reaches its maximum value unless the H counter has also reached its maximum value. This is to prevent the address to the ram changing once the final coefficient has been calculated.
* H counter never resets to 0, it remains at its maximum value until the circuit is reset.

### HOW ARE THE ADDRESSES COMPUTED FROM THE COUNTER VALUES? (examples given for matricies in project script 3x2 \* 2x3 = 3x3)

* **ROM\_A\_ADDRESS:**

****

* + The coefficients are stored in blocks of M in the ROM.
  + The offset is given by:

(note: M\_maximum is not the maximum counter value which starts from 0. M\_maximum is the width of matrix A starting from 1)

* + The address for ROM A is given by :
* **ROM\_B\_ADDRESS:**

****

* The coefficients are stored in blocks of N ( the matrix B width, not the the maximum N counter value)
* The offset is given by:

(note: N\_maximum is not the maximum counter value which starts from 0. N\_maximum is the width of matrix B starting from 1)

* The address for ROM B is given by :
* **RAM\_C\_ADDRESS:**

****

* + The coefficients are stored in blocks of N (matrix C width)
  + The offset is given by
  + The address of ROM A is given by :

## **QUestion 4:** What signals do you need to control the memories, the MACC unit, and the counters?

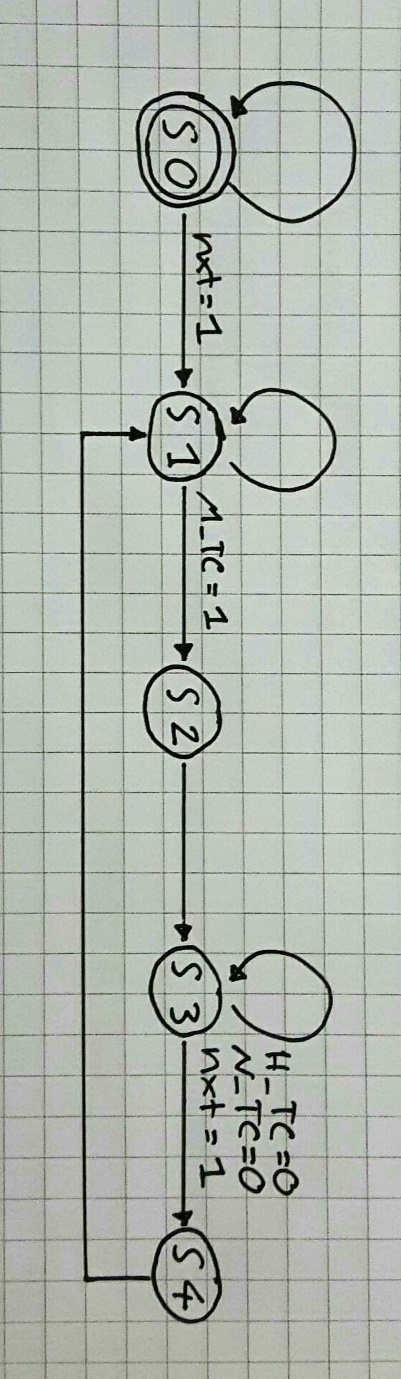
* Memory signals :
* *ROM\_A\_address (unsigned):* The address for the ROM storing the matrix A coefficients
* *ROM\_B\_address (unsigned):* The address for the ROM storing the matrix B coefficients
* *RAM\_C\_address (unsigned):* The address for the RAM storing the matrix C coefficients
* *RAM\_C\_write (std\_logic):* enables the RAM to store the value provided to its input.
* Counter signals:
  + *M\_counter\_value (unsigned):* Bus holding the value of the M counter
  + *N\_counter\_value (unsigned):* Bus holding the value of the N counter
  + *H\_counter\_value (unsigned):* Bus holding the value of the H counter
  + *M\_terminal\_count(std\_logic):* indicates that M has reached maximum value
  + *N\_terminal\_count(std\_logic):* indicates that M has reached maximum value
  + *H\_terminal\_count(std\_logic):* indicates that M has reached maximum value
  + *M\_counter\_enabled:* Enables the M counter
  + *M\_counter\_enabled:* Enables the M counter
  + *M\_counter\_enabled:* Enables the M counter
* MACC signals:
  + MACC\_enable: enables the MACC
  + MACC\_reset: resets the MACC

## **Question 5:** What is the behaviour of the state machine? How many states do you need? What are the control signals to be generated as a function of each state?

1. The state machine waits until the “next” input signal goes high.
2. The circuit starts to calculate the first coefficient of the C matrix. I does this by loading the values of the first A and B coefficients out of the ROM\_A and ROM\_B respectively. Those values are passed into the MACC, where they are multiplied together. The product of that multiplication is then added to the previous product ( for the first cycle this is 0 ). On the next clock cycle this sum then becomes the output of the MACC. Then M counter then increments by one, moving to the next coefficient in A and B. This continues until the M counter reaches its maximum value.
3. The MACC gets disabled ( so that it holds its value) and the RAM\_write\_enable goes high so that the output from the MACC can be loaded into the RAM.
4. Once the value has been written into the RAM it instantly appears at the output (due to the RAM being asynchronous).
5. The state machine then waits until the “next” signal goes high again to start the computation of the next coefficient. If however the maximum values of H and N have been reached the state machine will freeze and output the last coefficient of matrix C.

This state machine requires 5 states in total.

# state graph of the state machine, with a description of each state and a list of all the control signals.

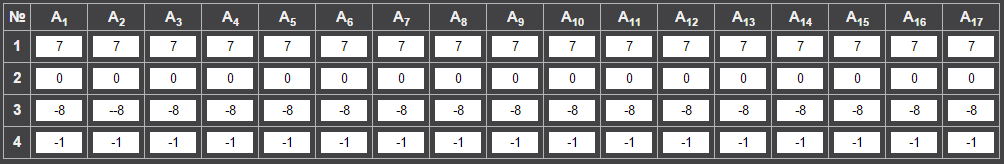


For address values see page 3.

# The values of the input matrices and of the product matrix

### N=17 N=8 H=4 datasize = 4

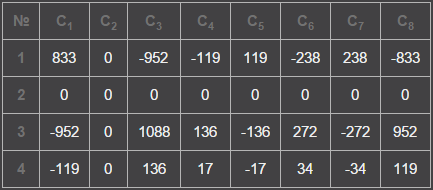
* ROM A:



* ROM B:

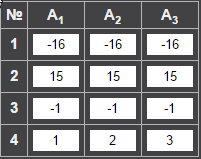
+

* RAM C (RESULT):

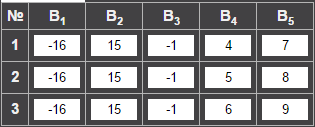


### M=3 N=5 H=4 dataSize = 5

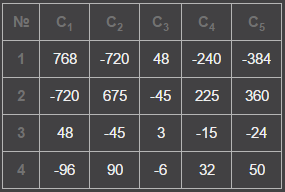
* ROM A:



* ROM B:



* RAM C (RESULT):



# VHDL code for EVERY component in the system.

### DIGENG.VHD (PACKAGE)

----------------------------------------------------

-- PACKAGE FOR DIGITAL ENGINEERING LABS

--

-- To use:

--

-- - Download file

-- - Use "Add copy" to add to Xilinx project

-- - Add "use work.DigEng.all" on top of entity

--

----------------------------------------------------

**package** DigEng **is**

**function** log2 **(**x **:** natural **)** **return** natural**;**

**function** size **(**x **:** natural **)** **return** natural**;**

**function** coeff\_C **(**data\_size **:** natural**;** M **:** natural**)** **return** natural**;**

**end** DigEng**;**

**package** **body** DigEng **is**

----------------------------------------------------

-- LOG BASE 2 FUNCTION

-- returns the ceiling of log base 2 of a (non-zero) integer

-- (1->0; 2->1; 3->2; 4->2; 5->3 ...)

--

-- This function is NOT SYNTHESIZABLE

-- should be used for indices, not circuit description

--

-- Examples:

-- - signal A : STD\_LOGIC\_VECTOR(log2(data\_size)-1 downto 0);

--

----------------------------------------------------

**function** log2 **(** x **:** natural **)** **return** natural **is**

**variable** temp **:** natural **:=** x **;**

**variable** n **:** natural **:=** 0 **;**

**begin**

**while** temp **>** 1 **loop**

temp **:=** temp **/** 2 **;**

n **:=** n **+** 1 **;**

**end** **loop** **;**

**if** **(**x **>** 2**\*\***n**)** **then**

n **:=** n **+** 1**;**

**end** **if;**

**return** n **;**

**end** **function** log2**;**

----------------------------------------------------

-- SIZE FUNCTION

-- returns the size of a vector that can encode a (non-zero) integer

-- (1->1; 2->2; 3->2; 4->3; 5->3 ...)

--

-- This function is NOT SYNTHESIZABLE

-- should be used for indices, not circuit description

--

-- Examples:

-- - signal A : STD\_LOGIC\_VECTOR(size(n)-1 downto 0);

--

----------------------------------------------------

**function** size **(** x **:** natural **)** **return** natural **is**

**variable** temp **:** natural **:=** x **;**

**variable** n **:** natural **:=** 0 **;**

**begin**

**while** temp **>=** 1 **loop**

temp **:=** temp **/** 2 **;**

n **:=** n **+** 1 **;**

**end** **loop** **;**

**return** n **;**

**end** **function** size**;**

----------------------------------------------------

-- coeff\_C FUNCTION

-- returns the size of the coefficient that results from

-- multiplying two coefficients of size data\_size

----------------------------------------------------

**function** coeff\_C **(**data\_size **:** natural**;** M **:** natural**)** **return** natural **is**

**variable** temp **:** natural**;**

**begin**

temp **:=** log2**(** **((**2**\*\*(**Data\_size**-**1**))\*\***2**)\***M **)** **+** 1**;**

**return** temp**;**

**end** **function** coeff\_C**;**

**end** DigEng**;**

## **Matrix\_Multiplier.VHD (TOP MODULE)**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.All;**

-- Top-level of the Matrix Multiplier design.

-- Instantiates the DataPath and Control\_logic

-- Design accepts two matrices with signed

-- coefficients (A & B) and multiplies them.

-- Coefficients are stored in two separate ROM's

-- and the product of them (matrix C) is stored in the RAM.

-- The design has a reset and next signal,

-- which starts the computation of the

-- next coefficient of matrix C.

**entity** Matrix\_Multiplier **is**

**GENERIC(** -- Matrix A width,Matrix B height

M **:** natural **:=** 17**;**

-- Matrix B width

N **:** natural **:=** 8**;**

-- Matrix A height

H **:** natural **:=** 4**;**

-- Size of Matrix A&B coefficient in bits

data\_size **:** natural **:=** 4**);**

**Port(** clk **:** **in** STD\_LOGIC**;**

nxt **:** **in** STD\_LOGIC**;**

rst **:** **in** STD\_LOGIC**;**

output **:** **out** signed**(**coeff\_C**(**data\_size**,**M**)-**1 **downto** 0**));**

**end** Matrix\_Multiplier**;**

**architecture** Behavioral **of** Matrix\_Multiplier **is**

--internal Address signals

**signal** ROM\_A\_address **:** unsigned**(**size**(**M**\***H**-**1**)** **-**1 **downto** 0**);**

**signal** ROM\_B\_address **:** unsigned**(**size**(**M**\***N**-**1**)** **-**1 **downto** 0**);**

**signal** RAM\_C\_address **:** unsigned**(**size**(**N**\***H**-**1**)** **-**1 **downto** 0**);**

--Control signals to other datapath components

**signal** RAM\_write\_enable **:** STD\_LOGIC**;**

**signal** MACC\_enable **:** STD\_LOGIC**;**

**signal** MACC\_reset **:** STD\_LOGIC**;**

-- De-bounced outputs

**signal** Deb\_next **:** STD\_LOGIC**;**

**signal** Deb\_reset **:** STD\_LOGIC**;**

**begin**

-- De-bouncer for the reset input

debouncer\_rst**:** **ENTITY** work**.**Debouncer

**PORT** **MAP(** clk **=>** clk**,**

sig **=>** rst**,**

Deb\_Sig **=>** Deb\_reset**);**

-- De-bouncer for the next input

debouncer\_nxt**:** **ENTITY** work**.**Debouncer

**PORT** **MAP(** clk **=>** clk**,**

sig **=>** nxt**,**

Deb\_Sig **=>** Deb\_next**);**

-- Instantiates the DataPath

-- Sub-Modules: ROM\_Matrix\_A.vhd, Rom\_Matrix\_B.vhd,

-- macc.vhd, RAM\_single\_port.vhd

DataPath **:** **ENTITY** work**.**Datapath

**GENERIC** **MAP(** M **=>** M**,**

N **=>** N**,**

H **=>** H**,**

Data\_size **=>** data\_size**,**

coeff\_C\_bits **=>** coeff\_C**(**data\_size**,**M**))**

**PORT** **MAP(** clk **=>** clk**,**

RAM\_write\_enable **=>** RAM\_write\_enable**,**

MACC\_enable **=>** MACC\_enable**,**

MACC\_reset **=>** MACC\_reset**,**

ROM\_A\_address **=>** ROM\_A\_address**,**

ROM\_B\_address **=>** ROM\_B\_address**,**

RAM\_C\_address **=>** RAM\_C\_address**,**

-- Output of the RAM

OUTPUT **=>** output**);**

-- Instantiates the Control\_logic

-- Sub-Modules Address\_generator.vhd Control\_Logic\_Counters.vhd fsm.vhd

Control\_logic **:** **ENTITY** work**.**Control\_logic

**GENERIC** **MAP(**M **=>** M**,**

N **=>** N**,**

H **=>** H**)**

**PORT** **MAP(** clk **=>** clk**,**

deb\_rst **=>** Deb\_reset**,**

deb\_nxt **=>** Deb\_next**,**

ROM\_A\_address **=>** ROM\_A\_address**,**

ROM\_B\_address **=>** ROM\_B\_address**,**

RAM\_C\_address **=>** RAM\_C\_address**,**

RAM\_C\_write\_enable **=>** RAM\_write\_enable**,**

MACC\_enable **=>** MACC\_enable**,**

MACC\_reset **=>** MACC\_reset**);**

**end** Behavioral**;**

## Datapath.VHD

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.All;**

**entity** Datapath **is**

**generic(**

M **:** natural **:=** 2**;** -- Matrix A&B width and height

N **:** natural **:=** 3**;** -- Matrix B width

H **:** natural **:=** 3**;** -- Matrix A height

Data\_size **:** natural **:=** 4**;** -- Size of Matrix A&B coefficient

coeff\_C\_bits **:** natural **:=** 8**);** -- Size of Matrix C coefficient

**Port(** clk **:** **in** STD\_LOGIC**;**

RAM\_write\_enable **:** **in** STD\_LOGIC**;**

MACC\_enable **:** **in** STD\_LOGIC**;**

MACC\_reset **:** **in** STD\_LOGIC**;**

ROM\_A\_address **:** **in** unsigned **(**size**(**M**\***H**-**1**)** **-**1 **downto** 0**);**

ROM\_B\_address **:** **in** unsigned **(**size**(**M**\***N**-**1**)** **-**1 **downto** 0**);**

RAM\_C\_address **:** **in** unsigned **(**size**(**N**\***H**-**1**)** **-**1 **downto** 0**);**

-- Output of the RAM

OUTPUT **:** **out** signed **(**coeff\_C\_bits**-**1 **downto** 0**));**

**end** Datapath**;**

**architecture** Behavioral **of** Datapath **is**

-- ROM A outputted coefficient

**signal** A\_coefficient **:** signed **(**Data\_size**-**1 **Downto** 0**);**

-- ROM B outputted coefficient

**signal** B\_coefficient **:** signed **(**Data\_size**-**1 **Downto** 0**);**

-- MACC output of Matrix C coefficient

**signal** MACC\_output **:** signed **(**coeff\_C\_bits**-**1 **Downto** 0**);**

**begin**

-- Instantiates ROM\_Matrix\_A.

ROM\_Matrix\_A **:** **ENTITY** work**.**ROM\_Matrix\_A

**generic** **MAP(**M **=>** M**,**

H **=>** H**,**

Data\_size **=>** Data\_size**)**

**PORT** **MAP(** Address **=>** ROM\_A\_address**,**

Matrix\_Coefficient**=>** A\_coefficient**);**

-- Instantiates ROM\_Matrix\_A.

Rom\_Matrix\_B **:** **ENTITY** work**.**Rom\_Matrix\_B

**generic** **MAP(**M **=>** M**,**

N **=>** N**,**

Data\_size **=>** Data\_size**)**

**PORT** **MAP(** Address **=>** ROM\_B\_address**,**

matrix\_coefficient**=>** B\_coefficient**);**

-- Instantiates the Multiply-Accumulate unit

macc **:** **ENTITY** work**.**macc

**GENERIC** **MAP(**data\_size **=>** Data\_size**,**

coeff\_data\_size **=>** coeff\_C\_bits**)**

**PORT** **MAP(** clk **=>** clk**,**

RST **=>** MACC\_reset**,**

EN **=>** MACC\_enable**,**

A **=>** A\_coefficient**,**

B **=>** B\_coefficient**,**

P **=>** MACC\_output**);**

-- Instantiates the RAM\_single\_port, used to store Matrix C

RAM\_single\_port **:** **ENTITY** work**.**RAM\_single\_port

**GENERIC** **MAP(**Maximum\_coefficient\_bit\_number **=>** coeff\_C\_bits**,**

N **=>** N**,**

H **=>** H**)**

**PORT** **MAP(** clk **=>** clk**,**

write\_enable **=>** RAM\_write\_enable**,**

data\_in **=>** MACC\_output**,**

Address **=>** RAM\_C\_address**,**

output **=>** OUTPUT**);**

**end** Behavioral**;**

### ROM\_Matrix\_A.VHD

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.ALL;**

**entity** ROM\_Matrix\_A **is**

**generic** **(** M **:** natural **:=** 2**;**

H **:** natural **:=** 3**;**

Data\_size **:** natural **:=** 4**);**

**Port(**Address **:** **in** unsigned **(**size**(**M**\***H**-**1**)-**1 **downto** 0**);**

Matrix\_Coefficient **:** **out** SIGNED **(**Data\_size**-**1 **downto** 0**));**

**end** ROM\_Matrix\_A**;**

**architecture** Behavioral **of** ROM\_Matrix\_A **is**

**type** ROM\_Array **is** **array** **(**0 **to** M**\***H**-**1**)** **of** SIGNED**(**Data\_size**-**1 **downto** 0**);**

**constant** Data**:** ROM\_Array **:=(**

--MAXIMUM POSITIVE COEFFICIENT VALUE

0 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**-- A1\_1

1 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

2 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

3 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

4 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

5 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

6 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

7 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

8 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

9 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

10 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

11 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

12 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

13 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

14 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

15 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**

16 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--A1\_17

--MINIMUM COEFFICIENT SIZE

17 **=>** **to\_signed(**0**,**Data\_size**),**--A2\_1

18 **=>** **to\_signed(**0**,**Data\_size**),**

19 **=>** **to\_signed(**0**,**Data\_size**),**

20 **=>** **to\_signed(**0**,**Data\_size**),**

21 **=>** **to\_signed(**0**,**Data\_size**),**

22 **=>** **to\_signed(**0**,**Data\_size**),**

23 **=>** **to\_signed(**0**,**Data\_size**),**

24 **=>** **to\_signed(**0**,**Data\_size**),**

25 **=>** **to\_signed(**0**,**Data\_size**),**

26 **=>** **to\_signed(**0**,**Data\_size**),**

27 **=>** **to\_signed(**0**,**Data\_size**),**

28 **=>** **to\_signed(**0**,**Data\_size**),**

29 **=>** **to\_signed(**0**,**Data\_size**),**

30 **=>** **to\_signed(**0**,**Data\_size**),**

31 **=>** **to\_signed(**0**,**Data\_size**),**

32 **=>** **to\_signed(**0**,**Data\_size**),**

33 **=>** **to\_signed(**0**,**Data\_size**),**--A2\_17

--MAXIMUM NEGATIVE VALUE

34 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**--A3\_1

35 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

36 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

37 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

38 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

39 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

40 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

41 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

42 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

43 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

44 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

45 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

46 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

47 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

48 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

49 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

50 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**--A3\_17

-- MINUS 1

51 **=>** **to\_signed(-**1**,**Data\_size**),**--A4\_1

52 **=>** **to\_signed(-**1**,**Data\_size**),**

53 **=>** **to\_signed(-**1**,**Data\_size**),**

54 **=>** **to\_signed(-**1**,**Data\_size**),**

55 **=>** **to\_signed(-**1**,**Data\_size**),**

56 **=>** **to\_signed(-**1**,**Data\_size**),**

57 **=>** **to\_signed(-**1**,**Data\_size**),**

58 **=>** **to\_signed(-**1**,**Data\_size**),**

59 **=>** **to\_signed(-**1**,**Data\_size**),**

60 **=>** **to\_signed(-**1**,**Data\_size**),**

61 **=>** **to\_signed(-**1**,**Data\_size**),**

62 **=>** **to\_signed(-**1**,**Data\_size**),**

63 **=>** **to\_signed(-**1**,**Data\_size**),**

64 **=>** **to\_signed(-**1**,**Data\_size**),**

65 **=>** **to\_signed(-**1**,**Data\_size**),**

66 **=>** **to\_signed(-**1**,**Data\_size**),**

67 **=>** **to\_signed(-**1**,**Data\_size**),**--A4\_17

**others** **=>** **to\_signed(**0**,**Data\_size**));**

**begin**

-- Asynchronous read

Matrix\_Coefficient **<=** Data**(to\_integer(**address**));**

**end** Behavioral**;**

### Rom\_Matrix\_B.VHD

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.ALL;**

**entity** Rom\_Matrix\_B **is**

**generic** **(** M **:** natural **:=** 2**;**

N **:** natural **:=** 3**;**

Data\_size **:** natural **:=** 4**);**

**Port** **(** Address **:** **in** unsigned **(**size**(**M**\***N**-**1**)-**1 **downto** 0**);**

matrix\_coefficient **:** **out** signed **(**Data\_size**-**1 **downto** 0**));**

**end** Rom\_Matrix\_B**;**

**architecture** Behavioral **of** Rom\_Matrix\_B **is**

--ROM FOR A 3x2 matrix ( 2 rows and 3 columns)

**type** Rom\_Array **is** **array** **(**0 **to** M**\***N**-**1**)** **of** signed**(**Data\_size**-**1 **downto** 0**);**

**constant** Rom\_B **:** Rom\_Array **:=(**

0 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B1\_1

1 **=>** **to\_signed(**0**,**Data\_size**),**

2 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

3 **=>** **to\_signed(-**1**,**Data\_size**),**

4 **=>** **to\_signed(**1**,**Data\_size**),**

5 **=>** **to\_signed(-**2**,**Data\_size**),**

6 **=>** **to\_signed(**2**,**Data\_size**),**

7 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B1\_8

8 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B2\_1

9 **=>** **to\_signed(**0**,**Data\_size**),**

10 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

11 **=>** **to\_signed(-**1**,**Data\_size**),**

12 **=>** **to\_signed(**1**,**Data\_size**),**

13 **=>** **to\_signed(-**2**,**Data\_size**),**

14 **=>** **to\_signed(**2**,**Data\_size**),**

15 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B2\_8

16 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B3\_1

17 **=>** **to\_signed(**0**,**Data\_size**),**

18 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

19 **=>** **to\_signed(-**1**,**Data\_size**),**

20 **=>** **to\_signed(**1**,**Data\_size**),**

21 **=>** **to\_signed(-**2**,**Data\_size**),**

22 **=>** **to\_signed(**2**,**Data\_size**),**

23 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B3\_8

24 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B4\_1

25 **=>** **to\_signed(**0**,**Data\_size**),**

26 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

27 **=>** **to\_signed(-**1**,**Data\_size**),**

28 **=>** **to\_signed(**1**,**Data\_size**),**

29 **=>** **to\_signed(-**2**,**Data\_size**),**

30 **=>** **to\_signed(**2**,**Data\_size**),**

31 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B4\_8

32 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B5\_1

33 **=>** **to\_signed(**0**,**Data\_size**),**

34 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

35 **=>** **to\_signed(-**1**,**Data\_size**),**

36 **=>** **to\_signed(**1**,**Data\_size**),**

37 **=>** **to\_signed(-**2**,**Data\_size**),**

38 **=>** **to\_signed(**2**,**Data\_size**),**

39 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B5\_8

40 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B6\_1

41 **=>** **to\_signed(**0**,**Data\_size**),**

42 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

43 **=>** **to\_signed(-**1**,**Data\_size**),**

44 **=>** **to\_signed(**1**,**Data\_size**),**

45 **=>** **to\_signed(-**2**,**Data\_size**),**

46 **=>** **to\_signed(**2**,**Data\_size**),**

47 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B6\_8

48 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B7\_1

49 **=>** **to\_signed(**0**,**Data\_size**),**

50 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

51 **=>** **to\_signed(-**1**,**Data\_size**),**

52 **=>** **to\_signed(**1**,**Data\_size**),**

53 **=>** **to\_signed(-**2**,**Data\_size**),**

54 **=>** **to\_signed(**2**,**Data\_size**),**

55 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B7\_8

56 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B8\_1

57 **=>** **to\_signed(**0**,**Data\_size**),**

58 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

59 **=>** **to\_signed(-**1**,**Data\_size**),**

60 **=>** **to\_signed(**1**,**Data\_size**),**

61 **=>** **to\_signed(-**2**,**Data\_size**),**

62 **=>** **to\_signed(**2**,**Data\_size**),**

63 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B8\_8

64 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B9\_1

65 **=>** **to\_signed(**0**,**Data\_size**),**

66 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

67 **=>** **to\_signed(-**1**,**Data\_size**),**

68 **=>** **to\_signed(**1**,**Data\_size**),**

69 **=>** **to\_signed(-**2**,**Data\_size**),**

70 **=>** **to\_signed(**2**,**Data\_size**),**

71 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B9\_8

72 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B10\_1

73 **=>** **to\_signed(**0**,**Data\_size**),**

74 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

75 **=>** **to\_signed(-**1**,**Data\_size**),**

76 **=>** **to\_signed(**1**,**Data\_size**),**

77 **=>** **to\_signed(-**2**,**Data\_size**),**

78 **=>** **to\_signed(**2**,**Data\_size**),**

79 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B10\_8

80 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B11\_1

81 **=>** **to\_signed(**0**,**Data\_size**),**

82 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

83 **=>** **to\_signed(-**1**,**Data\_size**),**

84 **=>** **to\_signed(**1**,**Data\_size**),**

85 **=>** **to\_signed(-**2**,**Data\_size**),**

86 **=>** **to\_signed(**2**,**Data\_size**),**

87 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B11\_8

88 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B12\_1

89 **=>** **to\_signed(**0**,**Data\_size**),**

90 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

91 **=>** **to\_signed(-**1**,**Data\_size**),**

92 **=>** **to\_signed(**1**,**Data\_size**),**

93 **=>** **to\_signed(-**2**,**Data\_size**),**

94 **=>** **to\_signed(**2**,**Data\_size**),**

95 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B12\_8

96 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B13\_1

97 **=>** **to\_signed(**0**,**Data\_size**),**

98 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

99 **=>** **to\_signed(-**1**,**Data\_size**),**

100 **=>** **to\_signed(**1**,**Data\_size**),**

101 **=>** **to\_signed(-**2**,**Data\_size**),**

102 **=>** **to\_signed(**2**,**Data\_size**),**

103 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B13\_8

104 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B14\_1

105 **=>** **to\_signed(**0**,**Data\_size**),**

106 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

107 **=>** **to\_signed(-**1**,**Data\_size**),**

108 **=>** **to\_signed(**1**,**Data\_size**),**

109 **=>** **to\_signed(-**2**,**Data\_size**),**

110 **=>** **to\_signed(**2**,**Data\_size**),**

111 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B14\_8

112 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B15\_1

113 **=>** **to\_signed(**0**,**Data\_size**),**

114 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

115 **=>** **to\_signed(-**1**,**Data\_size**),**

116 **=>** **to\_signed(**1**,**Data\_size**),**

117 **=>** **to\_signed(-**2**,**Data\_size**),**

118 **=>** **to\_signed(**2**,**Data\_size**),**

119 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B15\_8

120 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B16\_1

121 **=>** **to\_signed(**0**,**Data\_size**),**

122 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

123 **=>** **to\_signed(-**1**,**Data\_size**),**

124 **=>** **to\_signed(**1**,**Data\_size**),**

125 **=>** **to\_signed(-**2**,**Data\_size**),**

126 **=>** **to\_signed(**2**,**Data\_size**),**

127 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B16\_8

128 **=>** **to\_signed((**2**\*\*(**Data\_size**-**1**))-**1**,**Data\_size**),**--B17\_1

129 **=>** **to\_signed(**0**,**Data\_size**),**

130 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**)),**Data\_size**),**

131 **=>** **to\_signed(-**1**,**Data\_size**),**

132 **=>** **to\_signed(**1**,**Data\_size**),**

133 **=>** **to\_signed(-**2**,**Data\_size**),**

134 **=>** **to\_signed(**2**,**Data\_size**),**

135 **=>** **to\_signed(-(**2**\*\*(**Data\_size**-**1**))+**1**,**Data\_size**),**--B17\_8

**others** **=>** **to\_signed(**0**,**Data\_size**));**

**begin**

matrix\_coefficient **<=** Rom\_B**(to\_integer(**Address**));**

**end** Behavioral**;**

### RAM\_single\_port.VHD

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.All;**

---------------------------------------------------------

-- MATRIX C RAM (single port)

-- Synchronous write

-- Asynchronous read

---------------------------------------------------------

**entity** RAM\_single\_port **is**

**generic(**

-- The maximum number of bits needed to store the result of multiplying

-- the two matricies together. SIGNED !

Maximum\_coefficient\_bit\_number **:** natural **:=** 4**;**

-- The width of matrix B (N) and the depth of matrix A (H), for matrix

-- multiplication to be valid N=H, hence only one generic for both values.

H **:** natural **:=** 3**;**

N **:** natural **:=** 3**);**

**Port** **(** clk **:** **in** STD\_LOGIC**;**

write\_enable**:** **in** STD\_LOGIC**;**

data\_in **:** **in** signed **(**Maximum\_coefficient\_bit\_number**-**1 **downto** 0**);**

-- The RAM has a depth of H\*N

-- The address bus size needed is size(N\*H) bits

Address **:** **in** unsigned **(**size**(**N**\***H**-**1**)-**1 **downto** 0**);**

output **:** **out** signed **(**Maximum\_coefficient\_bit\_number**-**1 **downto** 0**));**

**end** RAM\_single\_port**;**

**architecture** Behavioral **of** RAM\_single\_port **is**

-- RAM\_depth , width of matrix B times depth of matrix a gives the number

-- of coefficients that the resultant matrix will have.

**constant** RAM\_depth **:** natural **:=** N**\***H**;**

-- Introduced RAM\_width constant for readibility of code

**constant** RAM\_width **:** natural **:=** Maximum\_coefficient\_bit\_number**;**

-- Creating RAM array to store the coefficients of matrix C

**type** RAM **is** **array(**0 **to** **(**2**\*\*(**size**(**RAM\_depth**-**1**)+**1**)-**1**))** **of** signed**(**RAM\_width**-**1 **downto** 0**);**

-- Declare RAM

**signal** RAM\_matrix\_C **:** RAM **;**

**begin**

---------------------------------------------------------

-- Asynchronous read

---------------------------------------------------------

output **<=** RAM\_matrix\_C**(to\_integer(**Address**));**

---------------------------------------------------------

-- Synchronous write

---------------------------------------------------------

RAM\_write**:** **PROCESS(**clk**)**

**begin**

**if(rising\_edge(**clk**))** **then**

**if(**write\_enable **=** '1'**)** **then**

RAM\_matrix\_C**(to\_integer(**Address**))** **<=** data\_in**;**

**end** **if;**

**end** **if;**

**end** **process;**

**end** Behavioral**;**

### macc.VHD

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.ALL;**

--Multiply-Accumulate unit (MACC).

--Accepts two SIGNED inputs (A and B), multiplies

--them together, producing the AprodB signal.

--The output is accumulated with output P

--resulting in the ABsumP output.

**entity** macc **is**

**generic(** -- size in bits of coefficient of matrix A and B

data\_size **:** natural **:=** 4**;**

-- size (in bits of) coefficient of matrix C

coeff\_data\_size **:** natural **:=** 8**);**

**Port(** clk **:** **in** STD\_LOGIC**;**

-- Sets the value of the Flip-Flops in the MACC to 0

RST **:** **in** STD\_LOGIC**;**

-- Enables the MACC

EN **:** **in** STD\_LOGIC**;**

-- Coeff from matrix A

A **:** **in** SIGNED**(**data\_size**-**1 **downto** 0**);**

-- Coeff from matrix B

B **:** **in** SIGNED**(**data\_size**-**1 **downto** 0**);**

-- Output coeff. Stored in RAM

P **:** **out** SIGNED**(**coeff\_data\_size**-**1 **downto** 0**));**

**end** macc**;**

**architecture** Behavioral **of** macc **is**

-- (A\*B)+ P result

**signal** ABsumP**:** SIGNED**(**coeff\_data\_size**-**1 **downto** 0**);**

-- Output of the MACC unit

**signal** ACCout**:** SIGNED**(**coeff\_data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

-- Signal is used for testing

**signal** AprodB**:** SIGNED**(**coeff\_data\_size**-**1 **downto** 0**);**

**begin**

-- Sum and product calculated separately so AprodB

-- signal is displayed on ISim

AprodB **<=** **to\_signed(to\_integer(**A**)\*to\_integer(**B**),**coeff\_data\_size**);**

ABsumP **<=** AprodB **+** ACCout**;**

acc **:** **process** **(**clk**)**

**begin**

**if(rising\_edge(**clk**))** **then**

-- Pass the input to the output, on enable signal

**if** **(**en **=** '1'**)** **then**

ACCout **<=** ABsumP**;**

**end** **if;**

-- Reset the unit to 0

**if** **(**RST **=** '1'**)** **then**

ACCout **<=** **(others** **=>** '0'**);**

**end** **if;**

**end** **if;**

**end** **process** acc**;**

-- ACC output to port P

P **<=** ACCout**;**

**end** Behavioral**;**

## Control\_logic.VHD

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.All;**

-- Instantiates the Address\_generator, Control\_Logic\_Counters and fsm

**entity** Control\_logic **is**

**generic(** M **:** natural **:=** 2**;**

N **:** natural **:=** 3**;**

H **:** natural **:=** 3**);**

**Port(** clk **:** **in** STD\_LOGIC**;**

deb\_rst **:** **in** STD\_LOGIC**;**

deb\_nxt **:** **in** STD\_LOGIC**;**

OM\_A\_address **:** **out** unsigned**(**size**(**M**\***H**-**1**)-**1 **downto** 0**);**

ROM\_B\_address **:** **out** unsigned**(**size**(**M**\***N**-**1**)-**1 **downto** 0**);**

RAM\_C\_address **:** **out** unsigned**(**size**(**N**\***H**-**1**)-**1 **downto** 0**);**

RAM\_C\_write\_enable **:** **out** STD\_LOGIC**;**

MACC\_enable **:** **out** STD\_LOGIC**;**

MACC\_reset **:** **out** STD\_LOGIC**);**

**end** Control\_logic**;**

**architecture** Behavioral **of** Control\_logic **is**

-- signals for the counter values

**signal** M\_counter\_value\_temp **:** unsigned**(**size**(**M**-**1**)-**1 **Downto** 0**);**

**signal** N\_counter\_value\_temp **:** unsigned**(**size**(**N**-**1**)-**1 **Downto** 0**);**

**signal** H\_counter\_value\_temp **:** unsigned**(**size**(**H**-**1**)-**1 **Downto** 0**);**

-- signals for the terminal count signals of the counters

**signal** m\_terminal\_count **:** STD\_LOGIC**;**

**signal** n\_terminal\_count **:** STD\_LOGIC**;**

**signal** h\_terminal\_count **:** STD\_LOGIC**;**

-- enables the M counter in the Control\_logic\_counters

**signal** m\_counter\_enable **:** STD\_LOGIC**;**

**signal** n\_counter\_enable **:** STD\_LOGIC**;**

**signal** h\_counter\_enable **:** STD\_LOGIC**;**

**begin**

-- Instantiates the Address\_generator which generates an

-- address based on the current counter values

Address\_generator **:** **ENTITY** work**.**Address\_generator

**GENERIC** **MAP(**M **=>** M**,**

N **=>** N**,**

H **=>** H**)**

**PORT** **MAP(** M\_counter\_value **=>** M\_counter\_value\_temp**,**

N\_counter\_value **=>** N\_counter\_value\_temp**,**

H\_counter\_value **=>** H\_counter\_value\_temp**,**

ROM\_A\_address **=>** ROM\_A\_address**,**

ROM\_B\_address **=>** ROM\_B\_address**,**

RAM\_C\_address **=>** RAM\_C\_address**);**

-- Instantiates the Control\_Logic\_Counters. Three counters, M, N and H

Control\_Logic\_Counters **:** **ENTITY** work**.**Control\_Logic\_Counters

**GENERIC** **MAP(**M **=>** M**,**

N **=>** N**,**

H **=>** H**)**

**PORT** **MAP(** clk **=>** clk**,**

m\_enable **=>** m\_counter\_enable**,**

n\_enable **=>** n\_counter\_enable**,**

h\_enable **=>** h\_counter\_enable**,**

reset **=>** deb\_rst**,**

m\_value **=>** M\_counter\_value\_temp**,**

m\_terminal\_count **=>** m\_terminal\_count**,**

n\_value **=>** N\_counter\_value\_temp**,**

n\_terminal\_count **=>** n\_terminal\_count**,**

h\_value **=>** H\_counter\_value\_temp**,**

h\_terminal\_count **=>** h\_terminal\_count**);**

-- Instantiates the Finite State Machine

fsm **:** **ENTITY** work**.**fsm

**PORT** **MAP(** clk **=>** clk**,**

nxt **=>** deb\_nxt**,**

rst **=>** deb\_rst**,**

h\_tc **=>** h\_terminal\_count**,**

n\_tc **=>** n\_terminal\_count**,**

m\_tc **=>** m\_terminal\_count**,**

macc\_en **=>** MACC\_enable**,**

macc\_rst **=>** MACC\_reset**,**

m\_counter\_en **=>** m\_counter\_enable**,**

n\_counter\_en **=>** n\_counter\_enable**,**

h\_counter\_en **=>** h\_counter\_enable**,**

write\_en **=>** RAM\_C\_write\_enable**);**

**end** Behavioral**;**

### fsm.VHD

### Control\_Logic\_Counters.VHD

### Address\_generator.VHD

# self-checking VHDL testbench for the system

# output of the simulations

## computation cycle of one complete product matrix coefficient

## snapshot of the final value of every coefficient of the product matrix

# HDL synthesis