Digital Systems Lab Assignment-4

Name: Pranav Sutar

Roll no. - CS20B029

➢ OBJECTIVES:

1) To learn to perform Matrix Multiplication in Verilog.
2) To build a signed adder and subtractor in a single circuit for any n-bits number
3) To build a signed multiplier and unsigned multiplier in a single circuit and extend it to N bits
4) To build a circuit that accepts three inputs of N bits and performs addition and multiplication sequentially on them (N-bit Signed Multiplier and Adder Circuit(MAC)).
5) To learn to call one module in another module.

➢ Description

$$\begin{bmatrix} Y1 \\ Y2 \\ Y3 \\ Y4 \end{bmatrix} = \begin{bmatrix} A1 & A2 & A3 & A4 \\ A5 & A6 & A7 & A8 \\ A9 & A10 & A11 & A12 \\ A13 & A14 & A15 & A16 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \\ X3 \\ X4 \end{bmatrix}$$

In mathematics, particularly in linear algebra, **matrix multiplication** is a binary operation that produces a matrix from two matrices. For matrix multiplication, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The resulting matrix, known as the **matrix product**, has the number of rows of the first and the number of columns of the second matrix. The product of matrices **A** and **B** is denoted as **AB**.

Matrix multiplication was first described by the French mathematician Jacques Philippe Marie Binet in 1812, to represent the composition of linear maps that are represented by matrices.

Applications of Matrix Multiplication:

Matrix multiplication is thus a basic tool of linear algebra, and as such has numerous applications in many areas of mathematics, as well as in applied mathematics, statistics, physics, economics, and engineering. Computing matrix products is a central operation in all computational applications of linear algebra.

It is used widely in such areas as network theory, solution of linear systems of equations, transformation of co-ordinate systems, and population modeling, to name but a very few.

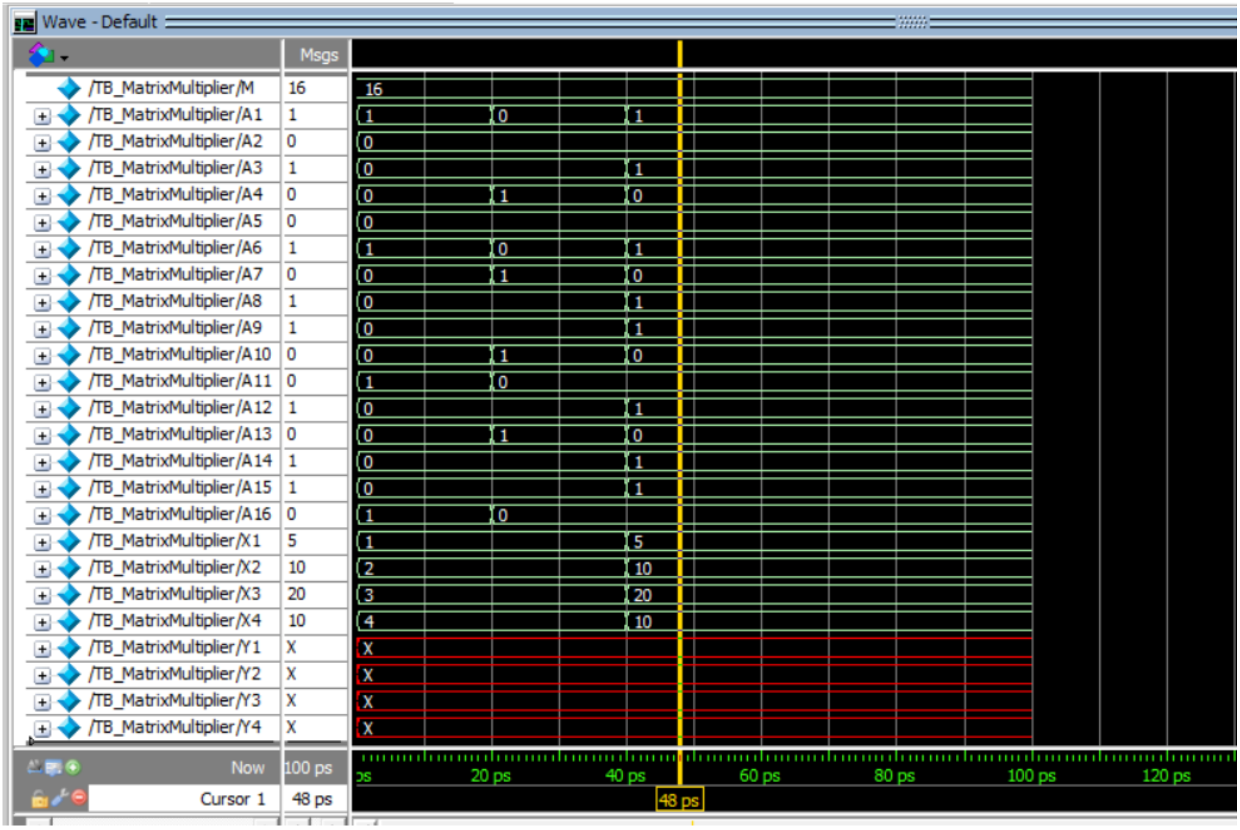
- ## Procedure

  The following modules are used in the main module to implement the circuit.

  - half_adder – It adds 2 bits and gives output in sum and carry form.
  - full_adder – It extends the above operation for three bits.
  - Nbit_add_sub – Adds the numbers when k = 0, and subtracts them when k = 0. The numbers are signed binary numbers.
  - comparator - It tells whether the bit is greater than or lesser than or equal to another bit
  - signed_multiplier – It multiplies two numbers of different size and and gives the corresponding output. It can perform signed as well as unsigned multiplication. Baugh-Wooley method is used to do signed multiplication.
  - MAC_signed – It peforms C = A.B+D operation, where all are in bit inputs in signed form.
  - MatrixMultiplier – Main module that contains MAC and it peforms Matrix multiplication operation

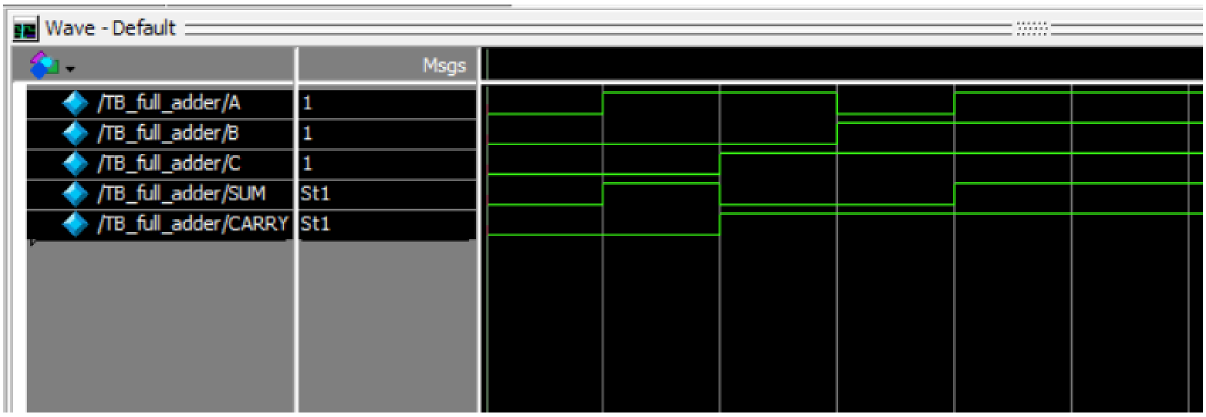
➢ **Result:**

Final TestModule:



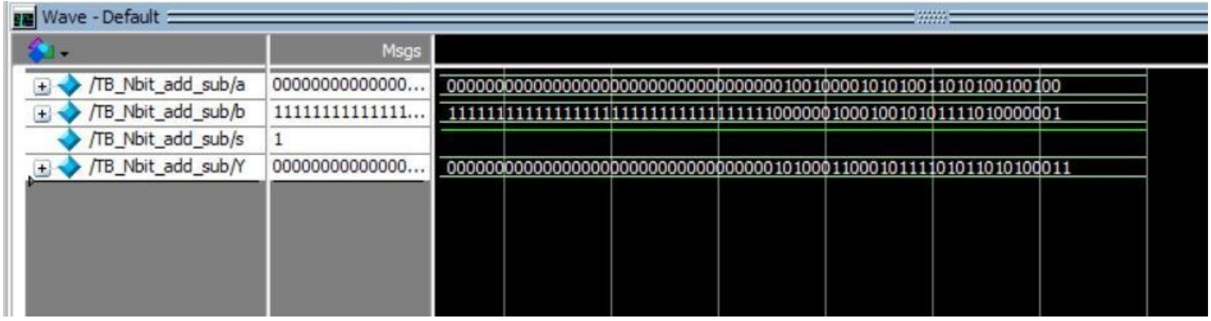
	Msgs
/TB_MatrixMultiplier/M	16
/TB_MatrixMultiplier/A1	1
/TB_MatrixMultiplier/A2	0
/TB_MatrixMultiplier/A3	1
/TB_MatrixMultiplier/A4	0
/TB_MatrixMultiplier/A5	0
/TB_MatrixMultiplier/A6	1
/TB_MatrixMultiplier/A7	0
/TB_MatrixMultiplier/A8	1
/TB_MatrixMultiplier/A9	1
/TB_MatrixMultiplier/A10	0
/TB_MatrixMultiplier/A11	0
/TB_MatrixMultiplier/A12	1
/TB_MatrixMultiplier/A13	0
/TB_MatrixMultiplier/A14	1
/TB_MatrixMultiplier/A15	1
/TB_MatrixMultiplier/A16	0
/TB_MatrixMultiplier/X1	5
/TB_MatrixMultiplier/X2	10
/TB_MatrixMultiplier/X3	20
/TB_MatrixMultiplier/X4	10
/TB_MatrixMultiplier/Y1	X
/TB_MatrixMultiplier/Y2	X
/TB_MatrixMultiplier/Y3	X
/TB_MatrixMultiplier/Y4	X
Now	100 ps
Cursor 1	48 ps

FULL Adder



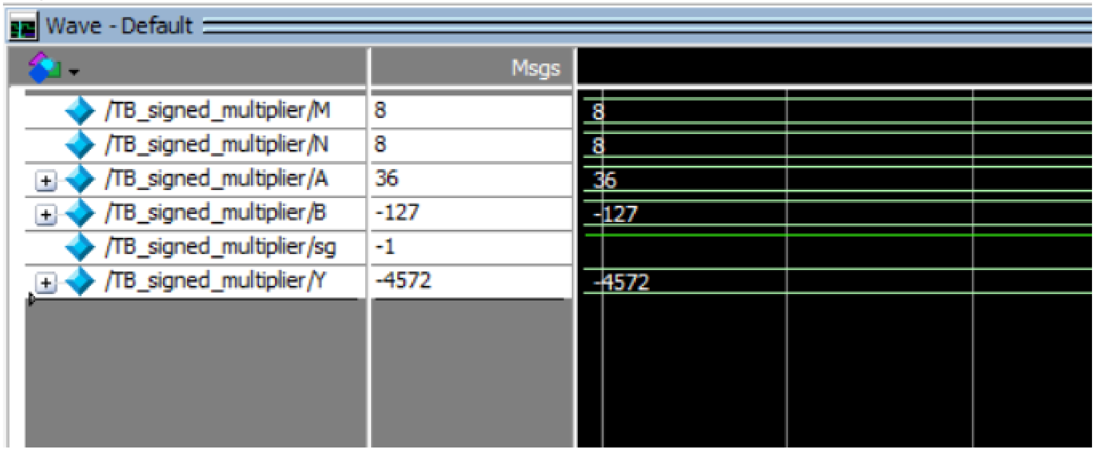
	Msgs
/TB_full_adder/A	1
/TB_full_adder/B	1
/TB_full_adder/C	1
/TB_full_adder/SUM	St1
/TB_full_adder/CARRY	St1

Signed Adder-Subtractor



```
/TB_Nbit_add_sub/a   00000000000000...   000000000000000000000000000000000000000100100001010100110101001001 00
/TB_Nbit_add_sub/b   11111111111111...   11111111111111111111111111111111111110000001000100101011110100000 01
/TB_Nbit_add_sub/s   1
/TB_Nbit_add_sub/Y   00000000000000...   00000000000000000000000000000000000010100011000101111010110101000 11
```

Signed Multiplier



```
/TB_signed_multiplier/M    8       8
/TB_signed_multiplier/N    8       8
/TB_signed_multiplier/A    36      36
/TB_signed_multiplier/B    -127    -127
/TB_signed_multiplier/sg   -1
/TB_signed_multiplier/Y    -4572   -4572
```