Nanyang Technological University
School of Electrical and Electronic Engineering

NTU-TUM MSc (IC Design) Programme
Digital IC Design Course: NM6008

Laboratory Manual

Module 5

Verilog Matrix Multiplier

Course Instructors: A/Prof Lim Meng Hiot
A/Prof Gwee Bah Hwee
Dr Cheng Deruo
Cheng Juncheng
Sheng Shirui

2nd – 13th October 2023

Section 1: Introduction

This tutorial style laboratory manual document is meant for NTU-TUM Master of Science (MSc) course – Integrated Circuit (IC) Design, NM6008. It builds upon the foundation and understanding acquired from going through the hands-on of earlier modules (Modules 3 and 4) by engaging students in the design and implementation of a practical digital system. The preparation of the contents of this manual benefits from the contributions of the following individuals:

Assoc Prof Lim Meng Hiot
Assoc Prof Gwee Bah Hwee
Dr Chong Kwen Siong
Mr Farid Ahamed
Mr Ne Kyaw Zwa Lwin

The following sub-sections state the objectives, the design tools/process used and the experimental design information of this module – *Verilog Matrix Multiplier*.

[Important Note: It is a pre-requisite that you have gone through the tutorials in Module 3 and Module 4. You are expected to refer to the laboratory manual documents of the Modules 3 and 4 for relevant information, if necessary.]

1.1 Objectives:

Module 5 pertains to design and implementation of a small system consisting of a state machine and two functional blocks. The specific objectives include:

(i) To reinforce students' understanding and familiarization with IC design methodologies by repeating all the design flows/steps emphasized in the Modules 3 and 4;

(ii) To apply their knowledge in the implementation of a digital system – hence any system (small or large) could be built in a similar manner.

1.2 Design Information

1.2.1 General Information – Matrix Multiplication

One of the most common mathematical operations in digital signal processing algorithms is matrix multiplication. Typical applications requiring matrix multiplication include Fast-Fourier Transform (FFT), image processing, etc.

Consider a simple matrix multiplication as follows. A 1×2 vector is multiplied by a 2×1 vector, the result of the multiplication is achieved as below:

$$[A1 \quad A0] \times \begin{bmatrix} B1 \\ B0 \end{bmatrix} = A1 \cdot B1 + A0 \cdot B0$$

Students are expected to realize a hardware circuit to perform such matrix multiplication.

1.2.2 Design Example – A Matrix Multiplier

Fig. 2 shows the primary inputs and outputs of the Matrix Multiplier; please use "MatrixMultiplier" as he top module name. Consider first the 5 inputs, namely *CLK*, *NRST*, *START*, *A*, and *B*, and their associated function(s). The *CLK* signal triggers the Matrix Multiplier to perform its operation (see more details later), and the *NRST* signal is an asynchronous input signal to reset the flip-flops. The *START* signal enables the matrix multiplication operation. Once the *START* signal is asserted, the Matrix Multiplier will perform the multiplication operation continuously, and such operation can only be interrupted by resetting the Matrix Multiplier. The *A* and *B* signals are the signed 8-bit input signals. The vectors A1 and A0 share the same A[7:0] signal bus, and hence the A1 and A0 signals are transferred sequentially. The same principle applies to the B[7:0] signal bus.

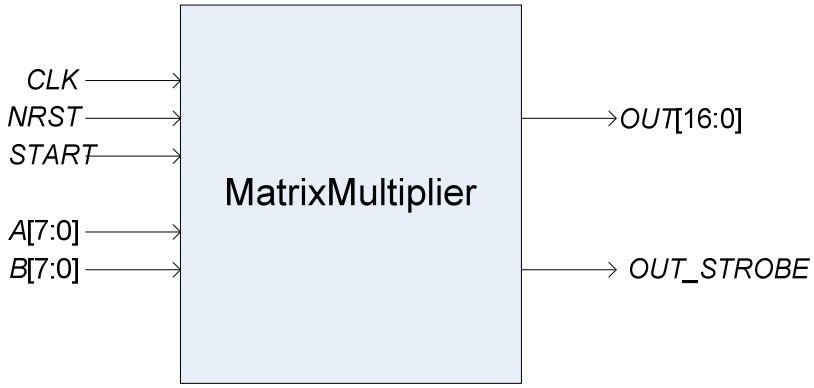

Fig. 2 The primary inputs and outputs of the Matrix Multiplier

Consider now the 2 outputs, namely *OUT* and *OUT_STROBE*. The *OUT* signal bus has 17 bits, generating the outcome of the matrix multiplication. The *OUT_STROBE* signal will be asserted '1' to indicate that the *OUT* signal is now valid/ready. Conversely, the *OUT_STROBE* signal remains '0', indicating that the *OUT* signal is not valid/ready.

Fig. 3 further shows a possible architecture of the Matrix Multiplier. There are three building blocks, the Multiplier, the Accumulator and the State Machine. The State Machine generates the three control signals (*EN*1, *EN*2 and *EN*3) to enable the data transfer between/within the Multiplier and the Accumulator.

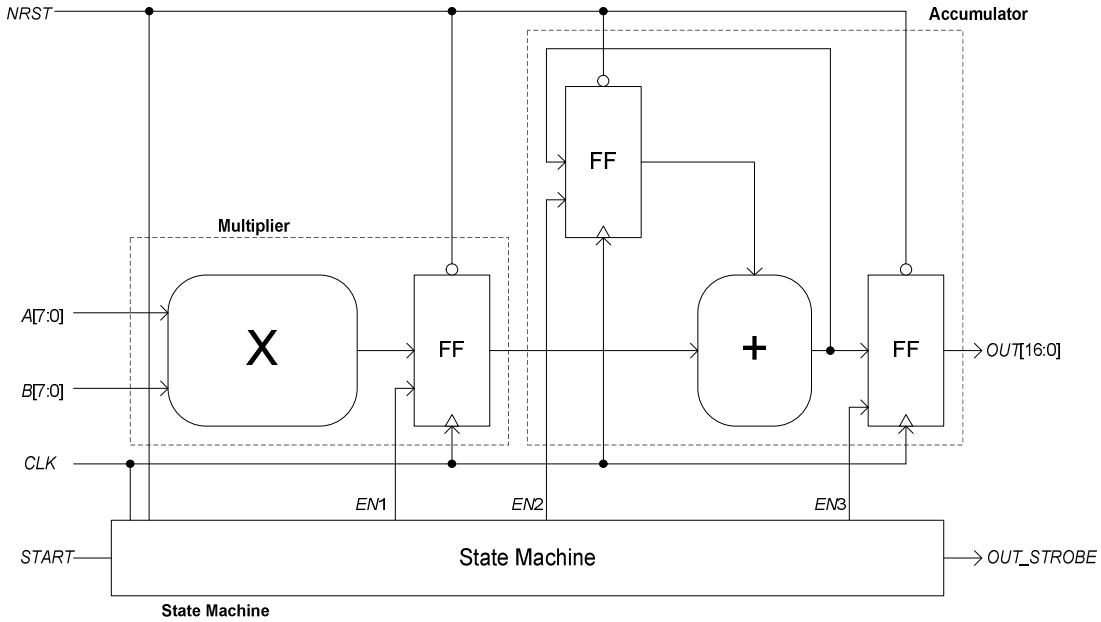

Fig. 3 A possible architecture of the Matrix Multiplier.

Students are expected to follow the architecture shown in Fig. 3 to realize the Matrix Multiplier.

1.2.3 Tips

(i) The signed multiplication can be modelled using the following code (where the *P* is output, and *A* and *B* are inputs)

```
assign P = $signed(A) * $signed(B);
```

(ii) For simplicity, all interfaced signals between the three building blocks can be registered (i.e. generated as sequential logic as opposed to combinational logic).

(iii) Signed extension is required for the Accumulator. For example, if the signal *D* is 17 bits and the signal *C* is only 16 bits, an extra bit can be added to *C* to make the final signal to be 17 bits.

```
assign Q = $signed({C[16], C}) + $signed(D);
```

Section 2: Hands-on

Based on the information in Section 1.2, you are required to carry out the following tasks.

1. Create a behavioural Verilog code for a matrix multiplier design and subsequently develop a test bench Verilog code to simulate the design. Ensure that you provide the input test patterns that you wish to be simulated in the file "inputvector.txt".
2. Simulate and observe the output waveforms obtained.
3. Synthesize the behavioural code to obtain a structural design of the matrix multiplier circuit based on the AMS technology libraries.
4. Perform post synthesis simulation(s) to visualize the behaviour of the circuit, taking into consideration the effect of the various gate delays based on the AMS technology libraries.
5. Perform a power simulation to estimate the overall dynamic as well as leakage power consumed by the overall circuit realized.
6. Submit a very (brief) report to the Instructor.