A PROGRAMMABLE SYSTOLIC ARRAY BASED MULTIPLIER DESIGN

Roderick Yap[1], Alexis Ayroso, Paula Marie Aureo, Martine Gabrielle De Leon
Department of Electronics and Communications Engineering
De La Salle University, Manila
2401 Taft Ave., Malate, Manila 1004, Philippines
[1] *roderick.yap@dlsu.edu.ph*

ABSTRACT

Systolic array is a popular method of implementing hardware-saving multiplier architecture. The multiplicand and multiplier bits are fed serially following a specific pattern of entry. In this paper, a programmable systolic array multiplier is implemented in gate level. The circuit is equipped with select pins which allow the user to set the number of bits for both the multiplicand and the multiplier from 1 bit to 8 bits. Though the processing of data is done serially, user feeding of the data is done in parallel. The product bits though generated serially will also be made available in parallel. To allow the user to save time in generating the product bits, a programmable output is provided. This output asserts high as soon as the product bits are completed.

Keywords: *systolic array, multiplier*

INTRODUCTION

Systolic array has found applications in many arithmetic operations. It consists of interconnected processing element with each element performing a specific operation [1]. These processing elements are connected together to form an array of data processors. It has been used in various types of applications ranging from computer architecture, parallel computing, and even in multiplication algorithms. In [3], bit-level systolic array for FIR digital filter with AND-based bit-serial pipelined multiplier was designed. In [4], a one dimensional systolic array was used for implementing FFT algorithm. Key features of systolic array include modular expandability, simple and regular data and control flows, use of simple and uniform cells and elimination of global broadcasting [1]. If there is one popular application of systolic array, then it has to be for multiplier algorithms. In [2], a hardware saving systolic array is presented for multiplier architecture design. It has found usefulness for large array processors. For an n bit multiplier, the number of processing element is n. Alternatively, Systolic array based integer multiplier can be of a single dimension with all signals of the multiplier flowing from input to output unidirectionally [1].

Design

The design starts with building the systolic array architecture to process the multiplier input bits. Fig.1 shows the general block diagram for the systolic array architecture. PE represents the processing element that is supposed to produce the product bits through stages of processing. R blocks represent shift registers. Inside the PE block is a full adder . The output C of the PE represents the output carry bit of the full adder. Circuit for the processing element is shown in Fig. 2. The processing element primarily consists of full adder

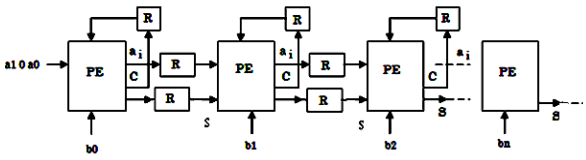


Figure 1 Systolic Array Architecture

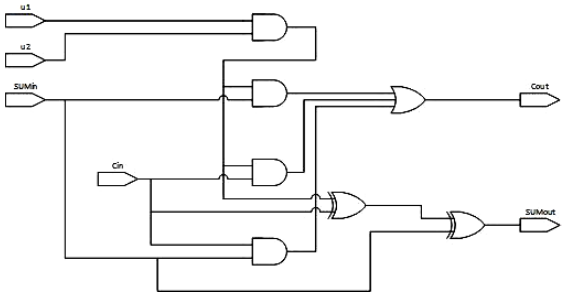


Figure 2 The Processing Element Circuit

Figure 3 shows the general block diagram of the programmable multiplier. The multiplier bits though fed in parallel are actually converted to serial bits which serve as inputs to the systolic array. Starting from clear setup, the formula block is set for MxN multiplication. The two inputs to the multiplier are assigned as “a” and ” b”.

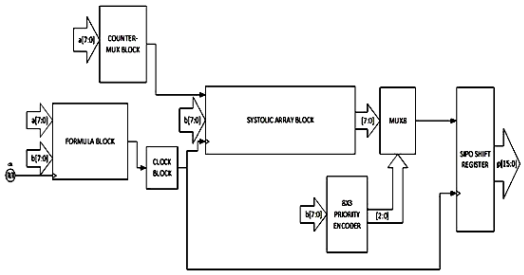


Figure 3 The General Block Diagram

The multiplicand, a, is entered bit by bit into the counter-mux block. The counter-mux block is responsible for making the multiplicand bits be fed serially into the systolic array block. Each bit of the serial multiplicand is placed alternately with a zero bit with the least significant bit (LSB) entering the systolic array first, just like in [2]. Placing a zero in between the multiplicand bits will ensure proper synchronization for the inputs entering the systolic array. The multiplier, b, is fed bit by bit into the systolic array block or more specifically, each bit of the multiplier is fed into each corresponding processing element with the LSB being fed into the first processing element while the second LSB of the multiplier is fed into the second processing element, and so on. Each of the output of the eight processing elements in the systolic array block is fed into an 8-bit multiplexer (MUX8 block). Moreover, the multiplier bits are also fed into an 8x3 priority encoder whose output will serve as the select lines for the previously mentioned 8-bit multiplexer. The 8x3 priority encoder and 8-bit multiplexer are the ones responsible for selecting the product of the multiplication performed in the systolic array block. Through the two blocks, the resizable feature of the binary multiplier is realized because the multiplexer can immediately choose the main output of the systolic array even before the eighth or last processing element if there are lesser bits to be multiplied for both NxN multiplication and MxN multiplication. The 8x3 priority encoder detects length of the multiplier which will then help determine which processing element would the multiplexer get the main output from. The formula block determines the number of clock cycle at which the clock of the whole circuit should stop based on the bit lengths of the multiplicand and multiplier. It also provides the stop bit for the whole circuit which that will signify that the binary multiplier is already finished multiplying. If X represents the number clock cycles to be computed by the formula block, then:

$$X = 2M + 3N - 1 \qquad (1)$$

Where M represents the number of multiplicand bits, N represents the number of multiplier bits. The above formula is changed to 2M if one of the factors is just a single bit. The output of the formula block will serve as the input of the clock block. The clock block's output will serve as the clock input for the systolic array and SIPO shift register blocks. Once the formula block has performed the necessary calculations, it will prompt the clock block to stop at the calculated stopping time, so the systolic array block will also stop, thus no further multiplications may be performed. Since the output coming from the multiplexer is still serial and is padded with a zero bit in between the main output bits, a Serial-In Parallel-Out shift register was used to separate the main output bit by bit and to eliminate the zero bits in between the output bits. The shift register, whose clock is controlled by the clock block, will also store each bit of the main output. Figure 4 shows the design of the formula block.

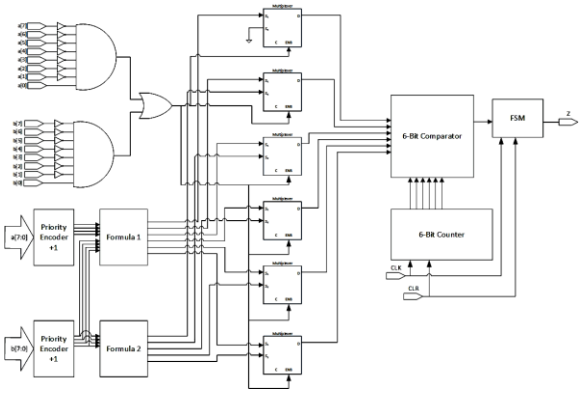


Figure 4 Formula Block

The formula block merely gives the number of clock pulse needed to generate the product bits. In order to control the multiplication process, an Finite State Machine (FSM) block is provided. Figure 5 shows the circuit for the FSM circuit. The FSM block monitors the counting after the reset state. A counter module counts the number of clock pulses that enter the system. Once the number of clock pulses matches the formula block output, the whole operation of the multiplier is stopped by the FSM circuit to avoid losing the product bits. Figure 6 shows the circuit used for comparing the number of clock pulses with the formula block output.

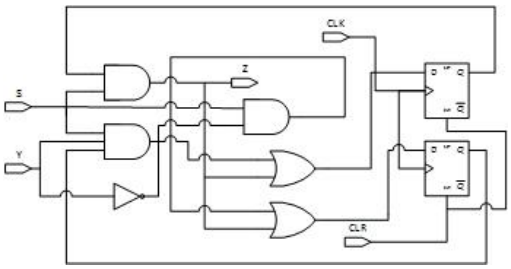


Figure 5 FSM Circuit

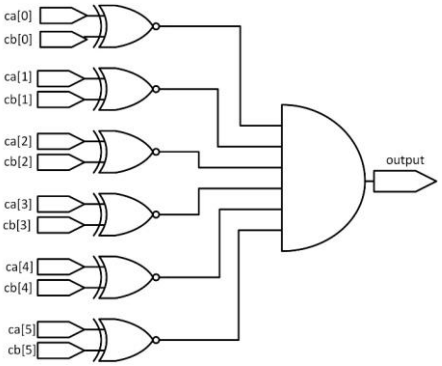


Figure 6 Comparator Circuit

Feeding of data is done using a multiplexer connected to a counter. This arrangement allows the feeding of the data input in parallel. The counter connected to the multiplexer will allow the parallel input data to be released serially into the pipelined systolic array cells. Each bit of the input is released alternately with zero. Figure 7 shows the circuit for the counter –multiplexer data feed.

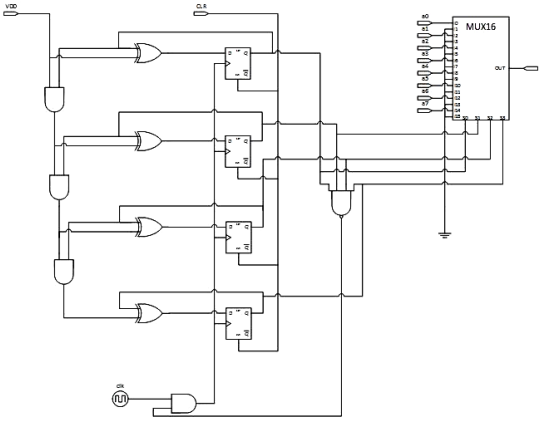


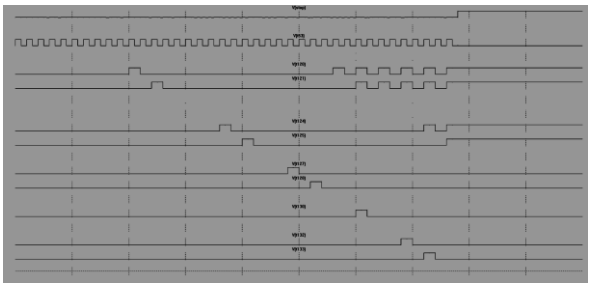
Figure 7 Counter-Multiplexer

The programmability of the system is in its ability to limit the number of clock pulses for multiplication based on the number of bits set by the user. The 8x3 priority encoder is responsible for determining the length of bits of the multiplier, b. Its inputs are tied with the multiplier input bits. The priority encoder will ignore the leading zeroes of the multiplier and determine its MSB. Its output will serve as the select line inputs for the 8-bit multiplexer which will select the processing element output from the systolic array. For example, if the multiplier input bits are 0000001x, the priority encoder will output a value of 001 which will then signal the 8-bit multiplexer to select the second processing element for its output.

DATA

The entire design was simulated for varying sizes of multiplicand and multiplier. Figure 8 shows 2 simulation results for 8 bit by 8 bit multiplication. Table 1 shows the other binary data used for simulation verification. All results met the expected product output.

11111111 x 11111111 = 1111111000000001



11101011x11111110 = 1110100100101010

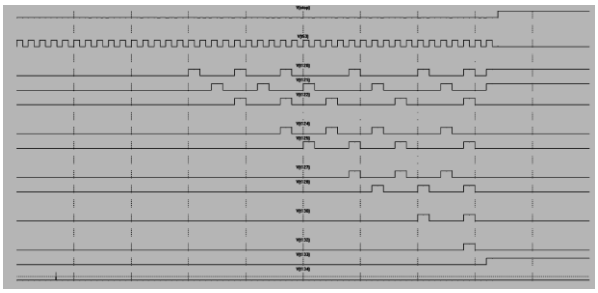


Figure 8 Some Simulation Results

Table 1 Simulation Result for Varying Number of Bits

1*1=1
10*10=100
10*11=110
11*11=1001
100*100=10000
101*110=11110
111*111=110001
1000*1000=1000000
1001*1010=1011010
11101*10001=111101101
100000x100000=10000000000

CONCLUSION

A CMOS-based resizable MxN binary multiplier with a maximum input capacity of 8 bits x 8bits using systolic arrays was successfully designed and simulated. The programmable systolic array allows the reconfiguration of the number of input bits for

NxN and NxM multiplication. The creation of a counter-multiplexer module allowed for the serial feeding of the multiplicand bits, which are entered bit by bit into the system along with the multiplier bits, into the systolic array. The systolic array was constructed like a pipeline system of processing elements which perform the multiplication operation. A multiplexer and a priority encoder were added in order to obtain the product much faster when the multiplier is of lower value. A circuit responsible for controlling and ending the operation of the systolic array at the right time based on the multiplicand and multiplier bits set was designed using modules like a comparator, a counter, multiplexer, priority encoders, and a controller circuit based on an FSM. A serial-in parallel-out (SIPO) shift register was also added to separate the product bits from the zeroes padded in between them coming from the systolic array, and to also display the product bit by bit instead of a serial output

RECOMMENDATION

The design was limited to a maximum of 8 bits by 8 bits. We recommend the expansion of the architecture to accommodate higher number of bits like 16 bit or 32 bits. The gate level design makes this paper promising for VLSI implementation. It is therefore recommended that a follow up research be done to implement the entire design in CMOS for possible IC implementation.

REFERENCES

[1] K. Obata, M. Tanaka et. al., “Single-flux-quantum integer mulitplier with systolic array structure”, Science Direct, Physica C 445-448 (2006) pp. 1014-1019

[2] L. Wang and I. Hartimo, “Systolic Array Multiplier, International ymposium on Speech, Image Processing and Neural Networks”., April, 1994

[3] Jae-Jin Lee; Gi-Yong Song, “Bit-level systolic array for FIR filter using AND-based bit-serial multiplier”, TENCON 2003. Conference on Convergent Technologies for the Asia-Pacific Region, Year: 2003, Volume: 1 Pages: 87 – 90

[4] Nandi, A.; Patil, S., “Performance Evaluation of One Dimensional Systolic Array for FFT Processor”, Signal Processing, Communications and Networking, 2007. ICSCN '07. International Conference on, 2007