Scientific Journal of Impact Factor(SJIF): 5.71

e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 13,Issue 1, January -2026

High-Speed MAC Unit Design Using Urdhva Tiryakbhyam Vedic Multiplication Algorithm

Minh Nguyen[1], Prof. Thao Tran[2], Prof. Linh Ho[3]

[1]*Research Scholar, Dept. of Computer Science, Hanoi University of Science and Technology, Vietnam*

[2]*Professor, School of Computing, Vietnam National University, Hanoi, Vietnam*

[3]*Associate Professor, Faculty of Engineering, Ho Chi Minh City University of Technology, Vietnam*.

Abstract: *This paper describes the implementation of a 32x32-bit multiply accumulate (MAC) unit designed using ancient Vedic mathematical techniques. This research work presents the efficiency of Urdhva Triyakbhyam Vedic method for multiplication which strikes a difference in actual process of multiplication itself. It enables the parallel generation of partial products and eliminates unwanted multiplication and addition steps. Multiply Accumulate unit is a key component in the most of the digital signal processors, in order to make a balance in the key performance characters such as speed, power and area, a gate level implementation of the design is adopted in the entire research work. An analysis of the best adder among some commonly available adders is carried out and the best adder is used for adding the partial product generated in the Vedic multiplication technique to reduce the combinational delay in the critical path. The proposed research work is coded in VHDL, and analysis in-terms of speed power and area is done using Xilinx ISE 14.6 tool.*

Key words: *Vedic mathematics, Vedic multiplier, Adders, Multiply Accumulate Unit, Verilog HDL*

I. INTRODUCTION

Multiplication is an important fundamental operation in arithmetic operations. Multiply and Accumulate (MAC) operations are used in many Digital signal processing (DSP) applications like FFT, DFT, convolution, and also in the arithmetic and logic unit of the microprocessors[8],[9]. In many DSP applications, the Multiply Accumulate component is a major contributing factor to the critical path delay and will affect the performance of the application. Low values of the critical path time delay and power consumption are the major specification for many applications. This paper describes a high speed, area efficient and low power 32x32 –bit multiply accumulate unit based on Vedic mathematics Multiplication operations performing in DSP applications, delay and throughput are two major specifications from a researcher's and designer's perspective. Time delay is the real delay of computing the algorithm and throughput is the measure of how many multiplication operations can be completed in a specified time. Minimizing power consumption and latency for digital system design involves optimization at all areas of the design [8] [9]. The optimization operation includes the best optimum algorithm for the operation based on specification, this being the highest level of design, then the circuit style, the topology and finally the technology used to implement the digital circuits. Most common multiplication algorithms followed in digital hardware are array multiplication algorithm and booth multiplication algorithm. Array multiplier is an efficient layout of combinational multiplier. Array multiplier circuit is based on add and shift algorithm. Partial products occurred during the multiplication of multiplicand with one multiplier bit and the partial products are added are shifted left or right according to the bit order and then added. (N-1) adders are required for N-bit multiplier. Booth multiplier is used for signed-number multiplication, which considers both positive and negative numbers in a same manner. It uses shift and add method to achieve the appropriate result. Each multiplier bit generates one multiple of the multiplicand which is to be added to the partial product. For N-bit multiplicand it requires N number of adders. Proposed paper uses Vedic-mathematics based

approach to reduce the number of partial products for multiplication, which in-effect reduces the number of adders. Vedic mathematics is the ancient Indian system of mathematics which is based on sixteen sutras and its sub-sutras mentioned in Atharva-Veda, and deals with various branch of mathematics such as arithmetic, algebra, geometry, trigonometry, conics, astronomy, calculus etc. The proposed architecture of 32x32-bit Multiply accumulate unit based on Vedic mathematics is shown in Fig1. The operand_1 and operand_2 are 32-bit data inputs. The 64-bit output of MAC is available in Result pin and carry from addition is available in Carry pin. The 64-bit adder performs addition of the result from multiplier to the value stored in the accumulator. To enhance the performance of MAC unit, adder for multiplication and accumulation is selected through a comparison of several adders.

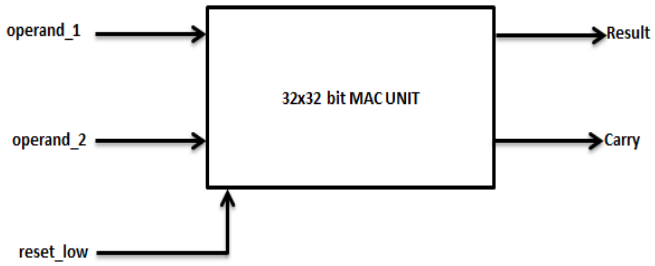

Fig1: Top level representation of MAC unit

II. VEDIC MATHEMATICS

The multiplier is based on an algorithm Urdhva Tiryakbhyam (Vertical and Crosswise) of ancient Indian Vedic mathematics. Urdhva Tiryakbhyam sutra is general multiplication formula applicable to all case of multiplication. It is based on a novel concept through which generation of all partial products can be done them; concurrent addition of these partial products can be done. Thus parallelism in generation of partial product is obtained by using Urdhva Tiryakbhyam sutra. The summation of the parallel product is done by using a high power carry save adder. The partial products and their sums are calculating in parallel blocks, so the multiplier path delay will not contribute to the critical path delay of the system. The strategy applied for developing a 32 x 32-bit Vedic multiplier is to design a 2 x 2- bit Vedic multiplier as a basic building module for the system. In the next stage of development a 4 x 4-bit multiplier is designed using 2 x 2-bit Vedic multiplier. Further in same manner 8 x 8, 16 x 16 and 32 x 32- bit Vedic multiplier is designed. For the partial product addition for all stages of development a fast carry look ahead adder is used.

A. 2x2 Vedic Multiplier

In 2 x 2-bit multiplier, the multiplicand has two bits each and result of multiplication is of four bits. Input ranges from "00" to "11" and the output lies in the set of "0000" to "1111". Fig 2 shows the stepwise multiplication of two binary numbers using Vedic mathematics technique.

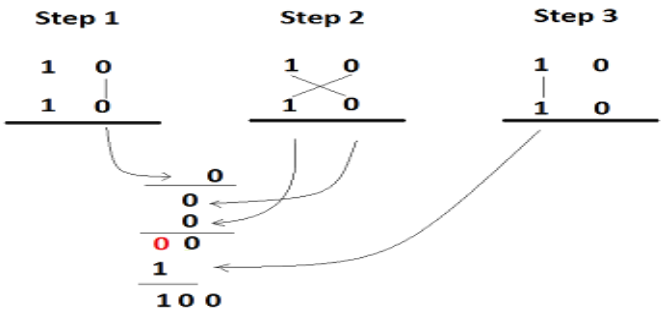

Fig2: Multiplication of "10" x "10"

The first step in multiplication is vertical multiplication of LSB of both multiplicands, and then second step is crosswise multiplication and additions of the partial products. Third step involves vertical multiplication of MSB of the multiplicand and addition with the carry propagated from step 2. Fig3 shows the hardware realization of 2x2 Vedic multiplier.

B. 4x4 Vedic Multiplier

The 4x4 multiplication is decomposed into four 2x2 multiplications performed in parallel. This mechanism reduces the number of stages for the multiplication and thus reduces the delay of the multiplier. Fig 4 shows the block level representation of the 4x4 Vedic multiplier.

The advantages of this mechanism is that larger bit streams (say N-bits) can be divided into (N/2=n) bit length, which can be further divided into n/2 bit streams and this can be continued till we reach the bit stream width of 2-bits, and the can be multiplied in parallel, thus providing an increase in speed of operation. The selection of the adder is based on a comparative study described in section III.

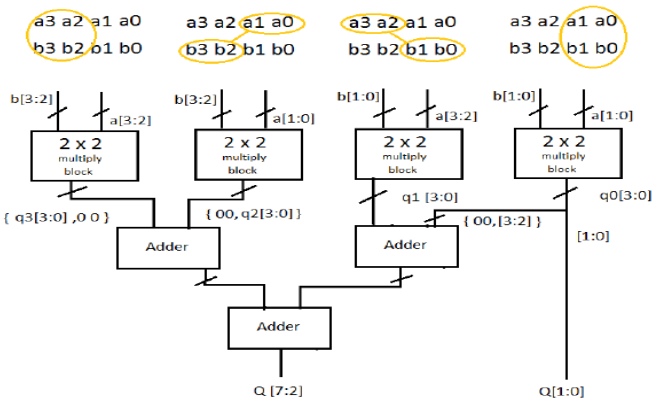

Fig3: Block level representation of 4x4 multiplier block

C. 8x8 Vedic Multiplier

The 8x8 multiplier is formed by using four, 4x4 multiplier blocks. Multiplicands are of bit size (n=8) where as the result is 16-bit size. The input is broken into smaller block of size n/2=4, for both inputs. The newly formed 4-bit data blocks are given as input to the 4x4 multiplier block, formed by 2x2 block. The results produced from the 4x4 multiplier blocks which is of 8-bit are given to the adder. Fig 4 represents the block level representation of the 8x8 multiplier block.

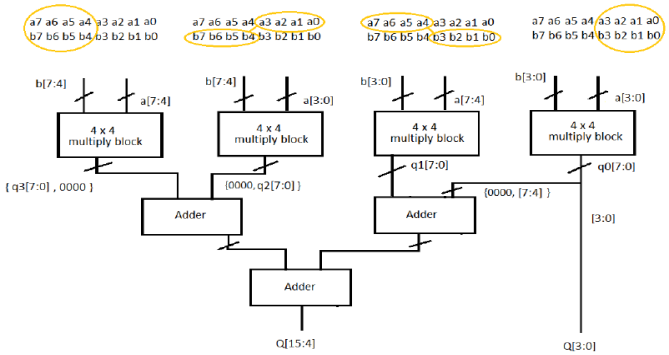

Fig4: Block level representation of 8x8 multiplier block

D. 16x16 Vedic Multiplier

The 16x16 multiplier is designed by using four 8x8 multiplier blocks. Both the multiplicands are of bit size (n=16) and the result is of 32-bit size. The input is broken into smaller block of size of n/2=8, for both the inputs. The newly formed 8x8 data blocks are applying to the input of 8x8 multiplier blocks. Fig5 repents the block level view of the 16x16 multiplier.

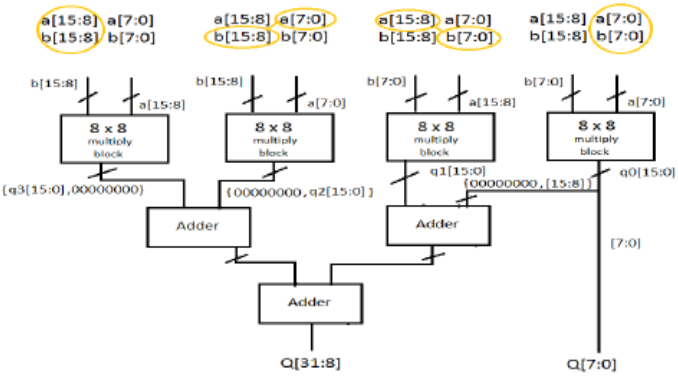

Fig5: Block level representation of 16x16 multiplier block

E. 32x32 Vedic Multiplier

The 32x32 multiplier is made by using four 16x16 multiplier blocks. The multiplicands are of bit size (n=32) where as the result is of 64-bit size. The input is broken into smaller block of size of n/2=16, for both the inputs. The newly formed 16x16 data blocks are applying to the input of 16x16 multiplier blocks. Fig6 repents the block level view of the 32x32 multiplier.

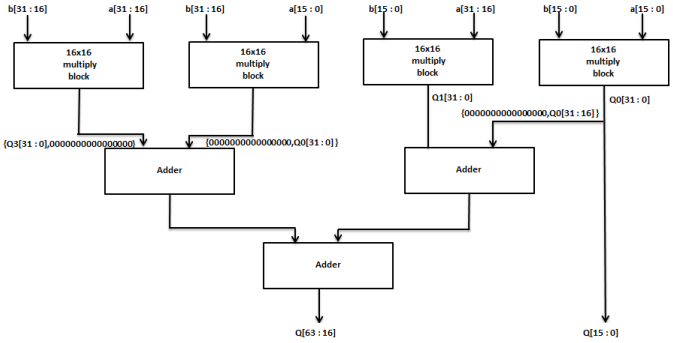

Fig 6: Block level representation of 32x32 multiplier block

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations

III. ANALYSIS OF ADDER

A. Carry Save Adder

Carry save adder is best suitable on adding more then 3- bits. The carry save adder is just a set of full adders and half adders. For an n-bit adder implementation, still carry is rippled to the next stage, of the same row, even though inputs to the lower next stage is ready, but kept in a wait state, until the sum and carry output didn't come from the above stage. This induces delay, now it can be optimized, if the carry out is passed diagonally to the lower next stage, instead of rippling to the next stage of the same row. Cary computation is not performed, but it is saved up to last row, where the results are obtained finally, in the last bottom row, carry is rippled however, but is significantly reduces the amount of delay occurred due to rippling operation. Fig 7 represents the carry save adder structure for a 4-bit addition.

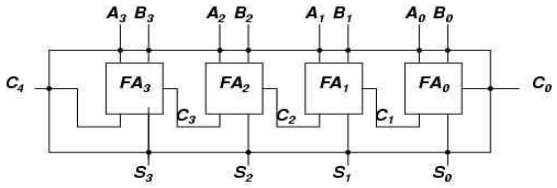

Fig7: Block level representation of carry save adder

The intermediate carry and sum is generated by using the half adder and it is given to full adder to perform the addition.

B. Carry Skip Adder

The carry skip adder reduces the delay as compared with carry look ahead adder and ripple carry adder. The carry skip adder divides the input word into blocks. Within each block, ripple carry adder is used to produce the sum bit and carry bit. The carry skip adder reduces the delay due to the carry computation i.e. by skipping over group of consecutive adder stages. If the input the individual adder blocks is different, sum will be generated and carry will not be computed and also the incoming carry is propagated to the next block. Also if both input to the blocks are zero, then incoming carry will not be propagated. Fig 8 represents the general block representation of the carry skip adder.

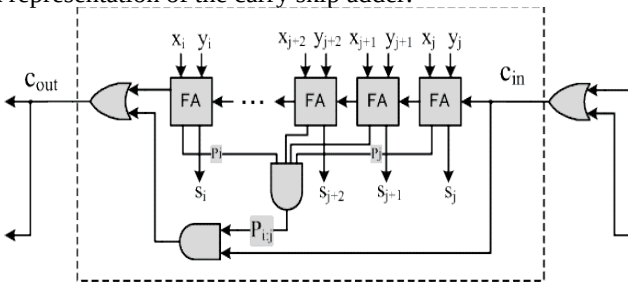

Fig 8: General block diagram of the carry skip adder

C. Ripple Carry Adder

The ripple carry adder is made using full adders only. The N-bit data input is directly applied to the N-full adders to perform the addition operation and the sum is generated in parallel. If the incoming is absent in the system, first full adder is replaced with the half adder. In ripple carry adder the carry output from each stage is rippled to the next stage. Fig9 shows the general block diagram of the 4-bit ripple carry adder.

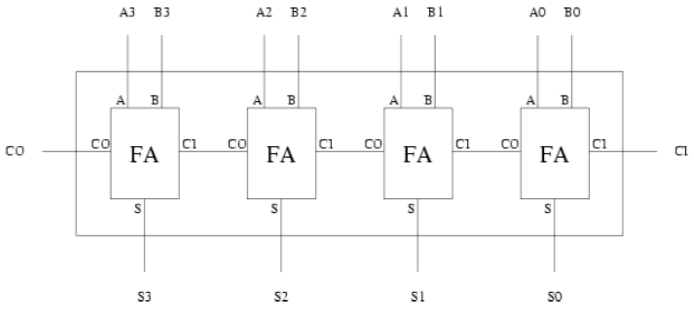

Fig 9: Block diagram of 4-bit ripple carry adder

D. Carry Look ahead Adder

The significant delay produced in ripple carry adder is a tradeoff; this can be minimized by computing the carry initially itself, as it will minimizes the wait for the carry at every stage. The carry look ahead adder is based on generate and propagate approach. Fig10 represents the logic equations used to represent the 4-bit carry look ahead adder.

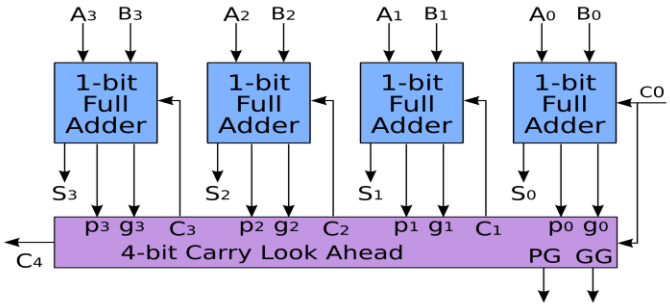

Fig 10: Block diagram of 4-bit ripple carry adder

IV. MULTIPLY ACCUMULATE UNIT

The multiply accumulate unit performs the multiplication and addition operation. For multiplication the proposed method uses Vedic multiplier and for addition, method uses carry look ahead adder. Fig11 shows the block level representation of the multiply accumulate unit.

The use of minim delay adder and a Vedic mathematics based multiplier for the multiply accumulate unit. The use of Vedic multiplier reduces significant amount of adders, required for the multiplier implementation. The fig 12 give the number of addition and multiplication required to implement the multiplier using conventional and Vedic multiplier approach.

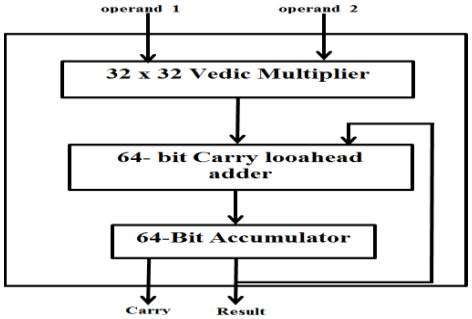

Fig11: Architecture of MAC unit

Input Bit Length	Number of calculations			
	Conventional		*Vedic*	
	M	A	M	A
2	4	2	4	1
3	9	7	9	5
4	16	15	16	9
8	64	77	64	53

M: Number of multiplications, A: Number of additions

Fig 12: Multipliers and additions required for implementation

V. CONCLUSION AND FUTURE SCOPE

Vedic multipliers are much faster than the conventional multipliers. This gives us method for hierarchical multiplier design. Therefore, the design complexity has reduced for inputs of large no of bits and modularity is increased. Urdhva tiryakbhyam, is algorithms, which can reduce the delay, power and hardware requirements for multiplication of numbers. the efficient implementation of the design various adders are studied, compared and among them carry look ahead adders is used for the final For implementation.

Future work includes the integration of the divider block, multiply and accumulate (MAC) unit, thereby making it into a Vedic Arithmetic and Logical unit (ALU). Future work includes the integration of the divider block, multiply and accumulate (MAC) unit, thereby making it into a Vedic Arithmetic and Logical unit.

REFERENCES

[1] T.T Hoang, M. Sjalander, “Double throughput Multiply Accumulate Unit for Flexcore processor enhancements”. IEEE International symposium on parallel and distributed processing, pp1-4, 2009
[2] Jagadguru Swami Sri Bharati Krishna Trithaji Maharaja, “Vedic Mathematics”,Motilal Banarsidas Publishers Pvt. Ltd Delhi, 2009
[3] A.D. Booth, “ A signed Binary multiplication Technique”, Qrt. J.Mech. App. Math., Vol 4., no.2, pp.236-240,1951.
[4] Akhter S., “VHDL implementation of a fast NxN multiplier based on Vedic mathematic”, pp. 472-475. ECCTD 2007.
[5] Shamsiah Suhaili and Othman Sidek, “ Design and implementation of reconfigurable alu on FPGA”, 3rd International Conference on Electrical & Computer Engineering ICECE 2004, 28-30 December 2004, Dhaka, Bangaladesh, pp.47-56.
[6] Tam Anh Chu, “Booth multiplier with low power high performace Input circuitry”, US patent, 6393454BL, May 21 2002
[7] Frank Marzona, “vedic mathematics: The Scientific Heritage of Ancient India”, Proceedings of the Pennsylvania State System of Higher Education Mathematics association Conference, held at Mansfield University 1997-volume one.
[8] Aneesh R, Jijuk, Sreekumari B, “Design and Implementation of Bluetooth MAC core with RFCOMM on FPGA”, proceedings of INDICON 2012.
[9] Aneesh R, Jijuk, “Design of FPGA based 8-bit RISC controller IP core using VHDL”, proceedings of INDICON 2012 5

