IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

VLSI Implementation of High-Performance Partial Product Reduction Vedic MAC

Korupoju Vikas
Kakatiya University College of Engineering, ECE
Warangal, India

Md. Asim Iqbal
Kakatiya University College of Engineering, Assistant Professor ECE
Warangal, India

Abstract:

The multiply-accumulate (MAC) architecture presented in this study makes use of Vedic mathematics to achieve excellent efficiency with minimal space requirements. While several MAC designs have implemented partial product reduction in multipliers to cut down on latency, this has introduced the additional challenge of dealing with overflow during the addition phase. This architecture, while efficient and advantageous to performance, is made in a sophisticated manner. To address this issue, we present a novel strategy that achieves state-of-the-art performance in terms of both space and delay. With the help of the ancient Indian mathematical treatise Urdhva-Tiryagbhyam sutra, we've developed a multiplier that saves the prior value in one register and uses the current value in the other, establishing a MAC architecture that saves valuable space by using only two input registers. The multiplier is an essential part of any high-performance MAC because it is used in digital signal processing systems and general processors. Finally, we will evaluate the efficiency of the proposed method versus the existing one.

Index Terms- Vedic Mathematics, Vedic Multiplier, Urdhva Tiryagbhyam, MAC.

Introduction

DSP is employed in numerous fields, including audio and speech coding, video and image processing, pattern recognition, sonar, and many others. DSP instructions can be implemented in real time on Very Large-Scale Integration (VLSI) chips, but only if they have a hardware architecture that can process input signals. The Multiplier Accumulator (MAC) unit is crucial to DSP applications due to the prevalence of multiply and accumulate operations in DSP computing. Many different Digital Signal Processing (DSP) applications are beginning to emerge, and the multiply and accumulate (MAC) Unit is at the centre of this. In addition, it supplements the microcontroller's signal-processing capabilities for a wide variety of applications, such as servo/audio control, etc. The Multiplier Acceleration Core (MAC) is an internal execution unit of the CPU that uses a three-tiered pipelined arithmetic structure to make the most of the CPU's eight 8-by-8 multipliers. Eight-bit opcodes can be used in the suggested design. There are primarily three procedures that can be carried out by any MAC unit: Various signing actions, include signed/unsigned operations, fixed-factor floating entry operations, and other similar tasks. Most

MAC devices are constructed by combining a multiplier with an accumulator. Due to the high efficiency requirements of DSP applications, carry pick and deliver keep adders are frequently employed in extreme instances. The multiplier receives the inputs from the memory unit, which it then multiplies and adds together. A MAC memory location is used to keep track of the final output from a MAC unit. All of these steps must happen within the same clock cycle for MAC to work [1].

The multiply-accumulate (MAC) unit is an elementary component in digital signal processing (DSP) [1]. Recently, real-time edge application development has emerged as a popular topic in the world of design [2,3]. As a result, there is a pressing need for low-power, high-efficiency MAC modules. The multiplier and the accumulator make up the two main components of a standard MAC unit (i.e., an accumulate adder). To prevent overflow, guard bits are utilised in a multiplier and adder of size (2N+-1) bits in an N-bit MAC unit (caused by long sequences of multiply-accumulate operations). Many earlier research [4–12] focused on optimising the multiplier and many others on optimising the adder. A variety of adder designs [8-10] have been presented, each with its own delay, area, and power trade-offs. It is also possible to create a wide range of MAC unit models by using a number of different multiplier and accumulator (adder) architectures. Reports [11,12] examine and contrast various MAC unit models in terms of delay, area, and power. Inside, the MAC unit is making progress, but just just. There's a good chance you've heard of the MAC unit before; it's a pretty variation on the standard primary multiplier unit found in almost all CPUs. It allows for a wide variety of high-performance digital signal processing (DSP) algorithms to run in a timeframe that is consistent with the needs of their respective applications.

The operation of multiplication is one of the most fundamental arithmetic operations[1]. Many digital signal processing (DSP) applications, including convolution, Fast Fourier Transform (FFT), filtering, and the arithmetic and logic unit of microprocessors, make use of Computation-Intensive Arithmetic Functions (CIAF) such Multiply and Accumulate (MAC) and inner product[2][3]. Multimedia applications like 3D graphics and signal processing systems rely heavily on the speed with which large numbers of multiplications may be executed[4]. Most digital signal processing (DSP) methods require a high-efficiency multiplier because multiplication is such a time-consuming operation. The amount of time needed to complete multiplication still plays a significant role in determining the instruction cycle time of a DSP processor.

Depending on the cost and transistor budget allotted for this operation, microprocessors accomplish multiplication in hardware and software in a number of different ways. In the early days of computing, complicated tasks were typically coded into software or the machine's micro-code. Formulation of a multiplier's design that uses only combinational logic to produce the product of two numbers As the need for more efficiency increases and the price of hardware continues to drop, it is increasingly conceivable that the multiplication will be implemented entirely in hardware. Most DSP algorithms rely on their multipliers for speedy computation, hence efficient multipliers are in high demand.

Existing Work

High power consumption and path latency are typical results of carry propagations of additions (including final additions in multiplications and additions in accumulations) in a typical MAC unit. To fix this issue, we have an incentive to shorten the carry propagation distances in the sum total and the sum total accumulation. To simplify the PPR procedure, We propose incorporating some of the additions (both the last addition and the accumulation) into it. This means that carry propagation distances can be cut down to size. In a MAC architecture, critical path delays (caused by the carry propagations) can be mitigated by postponing the addition and accumulation of higher priority bits until the PPR process of the next multiplication. We get one PPM from the PPG and another from the accumulation, and this is why our PPM (for the PPR process) is a hybrid of the two.

When using the Dadda tree algorithm, the final adder has only (2N-k-1) bits, where N is the number of bits in each input and k is the number of higher significance bits whose additions (accumulation) are skipped. Therefore, N = 4, as this is a 4-bit MAC. And here, for simplicity, we'll suppose that k = 3. As can be seen in Fig. 1(b), this means that the final addition can be performed using a 4-bit adder. It is important to keep in mind that the accumulation of bits with greater significance is not being done.

The values of these two product terms are sent to a -bit adder so that the total number of carries across the entire series of multiplication and accumulation operations may be calculated (accumulator). Figure 1(b) shows that the PPM generated by the product terms in the other columns (which includes the addition result of the (2N-k-1)-bit adder) are instead stored in register accumulation. The next PPM will incorporate the outcome of the next PPG as well as the present accumulation, so keep that in mind (i.e. the PPM created by the current accumulation).

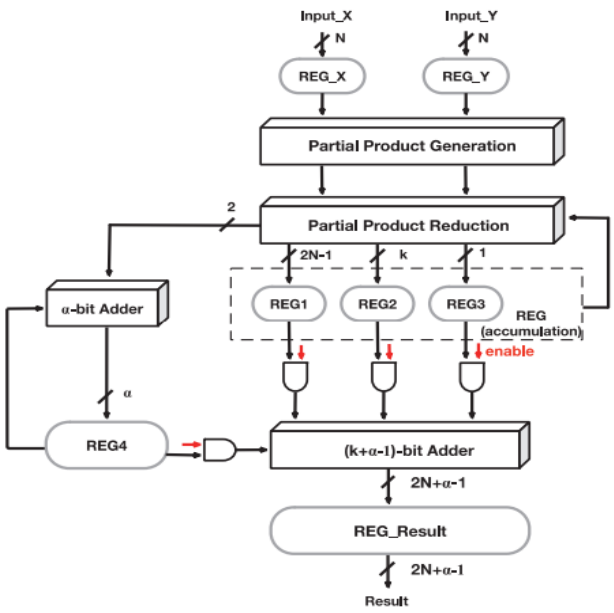

Fig. 1. The Existing MAC architecture

MAC protocol with two stages. In the first stage, we perform the PPG, PPR, (2N-k-1)-bit addition (representing some of the final addition), and -bit addition, with the PPM combining the PPG result with the accumulation result (for dealing with the overflow in the PPR process). Next, the (k+)-bit addition is carried out in the second step to generate the accumulation result. Below, we outline the key components of the suggested architecture. As a means of shortening carry propagations, we incorporate some changes into the PPR procedure. The PPR method uses a -bit adder to tally the amount of carries, which allows for handling of overflow. The second stage (of the full series of multiply-accumulate operations) is delayed until the final cycle (to save power) using the gating technique. Figure 1 depicts a MAC unit with a two-stage pipeline. Both the PPG and the accumulation contribute to our PPM, making it the sum of the two. The PPM for an unsigned MAC unit can be generated using a simple "AND" gate included directly into the PPG procedure. In order to create the signed PPM for a signed MAC unit, several PPG algorithms [25-27] have been developed, all of which take into account the effects of the sign bit. The signed PPM is generated by the PPG process in the proposed architecture using the Baugh-Wooley algorithm [25,26].

Proposed Work

Vedic mathematics is useful because it reduces the complexity of calculations used in Western mathematics to elementary levels. This is because the Vedic equations are rumoured to be based on the same natural principles upon which the human mind operates. To that end, Vedic Mathematics provides a set of guidelines for numerical computation that have been shown to increase efficiency. This area of study is fascinating, because it provides several useful techniques that can be used in other areas of engineering, including computing. The foundation of Vedic mathematics is a collection of sixteen Sutras (or aphorisms) that include topics like number theory, algebra, geometry, and more. Quicker mental arithmetic is the goal of these sutras. This list of Sutras is organised alphabetically, and their brief explanations follow. Various branches of applied mathematics, including trigonometry, plane geometry, spherical geometry, conic sections, differential and integral calculus, and more, can profit from these techniques and ideas.

The multiplication diagram shows how the integers on both sides of the line are multiplied and added

together with the carry from the previous operation. In addition to producing a carry, this also produces one of the result bits. The process continues when this carry is introduced in the subsequent phase. When a carry has multiple lines, the total number of carries is added up. The result of an operation is stored in the least significant bit (LSB) and the remainder is used as carry for the next operation. We'll start with the baseline assumption that the carry is 0. Everything is repeated, with the least important step taken first and the most important step taken last.

The product's least significant bit is calculated by multiplying the inputs' least significant bits (vertical). The multiplicand's least significant bit (LSB) is multiplied by the multiplier's highest-order bit (HMB), and the result is added to the multiplier's HMB times the multiplicand's LSB (crosswise). The second bit of the product is obtained by conducting a cross- and vertical-multiplication, adding the three bits of the two integers starting at the least significant place, and then adding the carry in the output of the subsequent stage sum. Next, the four bits are multiplied and added diagonally to derive the sum and carry. The next stage involves multiplying and adding three bits, excluding the LSB, and adding the carry once more. The same process is repeated until the MSB of the product is obtained by multiplying the two MSBs together.

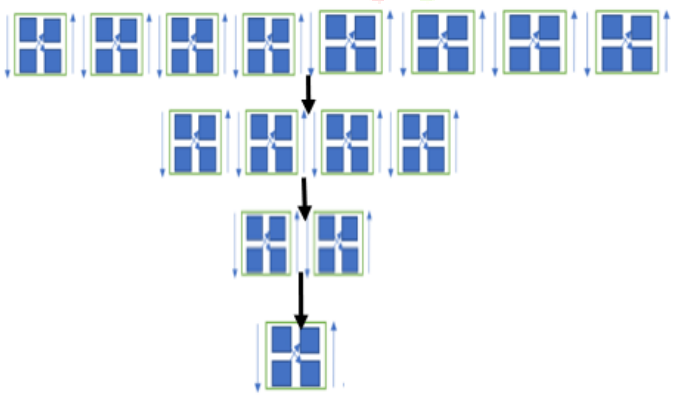

Figure.2 Multiplication method of Urdhva-Tiryakbhyam.

Figure-2 above takes into account the 4-bit binary values A0A1A2A3 and B0B1B2B3. All seven digits of the resulting result, R0R1R2R3R4R5R6R7, are recorded. The first operation is multiplying [A0, B0] and saving the result in register 0. The second step involves a complete adder being used to multiply [A0, B1] by [A1, B0], with the result being stored in R1 and the carry being passed on to the next step.

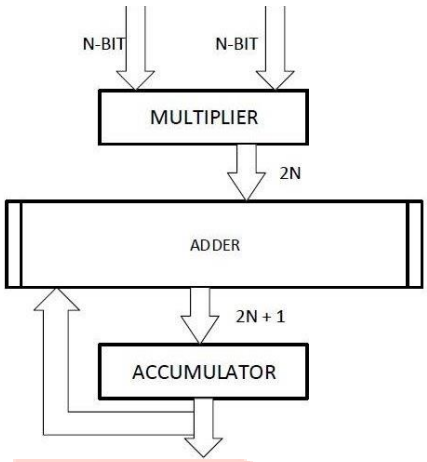

Fig. 3 MAC Architecture.

As soon as the multiplier's design is finished, it is added to the MAC chip. Given that we can deal with the resulting carry over, no more electronics are required.

Results

The Outcomes display the usage of both area and delay as it relates to the various FPGA families. The suggested Vedic provides simpler design options than the current standard of care.

Since we will be simulating in Xilinx ISE, we will ensure that our work does not exceed the allotted simulation time by counting the number of LUTs our project uses.

Area

```
Device utilization summary:
---------------------------

Selected Device : xa7a100tcsg324-2i


Slice Logic Utilization:
 Number of Slice Registers:              96  out of  126800     0%
 Number of Slice LUTs:                  749  out of  63400     1%
    Number used as Logic:               749  out of  63400     1%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    797
   Number with an unused Flip Flop:     701  out of    797    87%
   Number with an unused LUT:            48  out of    797     6%
   Number of fully used LUT-FF pairs:    48  out of    797     6%
   Number of unique control sets:         1

IO Utilization:
 Number of IOs:                         66
 Number of bonded IOBs:                 66  out of    210    31%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:               1  out of     32     3%
```

Delay

```
Minimum period: 10.046ns (Maximum Frequency: 99.541MHz)
Minimum input arrival time before clock: 0.831ns
Maximum output required time after clock: 0.761ns
Maximum combinational path delay: No path found
```

Parameters	Area	Delay/Efficiency
Existing work	123 LUT's	5.011ns
Proposed work	96 LIT's	10.046ns

Conclusion

This work introduces a two-stage pipeline MAC architecture with low power consumption and great efficiency for use in real-time DSP systems. We propose a system that streamlines and simplifies the PPR procedure. Urdhavatriyagbhyam, a performance, was studied for this purpose. Using a 16-by-16-bit Vedic multiplier significantly decreases the delay time, which boosts productivity and saves space. Along with saving time and space, simplifying the design of this MAC unit has its own set of benefits.

REFERENCES

[1] A. Abdelgawad, "Low Power Multiply Accumulate Unit (MAC) for Future Wireless Sensor Networks", in Proc. IEEE Int. Symp. Sensors Applications, Galveston, TX, 2013.

[2] H.O. Ahmed, M. Ghoneima and M. Dessouky, "Concurrent MAC Unit Design using VHDL for Deep Learning Networks on FPGA", in Proc. IEEE Int. Conf. Computer Applications and Industrial Electronics, Penang, Malaysia, 2018.

[3] V. Camus, C. Enz and M. Verhelst, "Survey of Precision-Scalable Multiply-Accumulate Units for Neural-Network Processing", in Proc. IEEE Int. Conf. Artificial Intelligence Circuits and Systems, Hsinchu, Taiwan, 2019.

[4] W.J. Townsend, E.E. Swartzlander, J.A. Abraham, "A Comparison of Dadda and Wallace Multiplier Delays", in Proc. SPIE Annual Meeting Optical Science and Technology, San Diego, CA, 2003.

[5] K.L.S. Swee and L.H. Hiung, "Performance Comparison Review of 32-Bit Multiplier Designs", in Proc. IEEE Intelligent and Advanced Systems, Kuala Lumpur, Malaysia, 2012, pp. 836-841.

[6] S. Asif and Y. Kong, "Design of an Algorithmic Wallace Multiplier using High Efficiency Counters", in Proc. IEEE Int. Conf. Computer Engineering & Systems, Cairo, Egypt, 2015, pp. 133-138.

[7] C.W. Tung and S.H. Huang, "Low-Power High-Accuracy Approximate Multiplier Using Approximate High-Order Compressors", in Proc. IEEE Int. Conf. Communication Engineering and Technology, Nagoya, Japan, 2019.

[8] C. Nagendra, M.J. Irwin and R.M. Owens, "Area-Time-Power Tradeoffs in Parallel Adders", IEEE Trans. Circuits and Systems -- II: Analog and Digital Signal Processing, vol.. 43, no. 10, pp. 689-702, Oct. 1996.

[9]. Amrita Nanda, "Design and Implementation of Urdhva-Tiryakbhyam Based Fast 8×8 Vedic Binary Multiplier" IJERT, ISSN: 2278- 0181,Vol. 3 Issue 3, March – 2014.

[10]. Poornima M, Shivaraj Kumar Patil, Shivukumar, ShridharKP, Sanjay H, "Implementation of Multiplier Using Vedic Algorithm", JITEE, ISSN:-2278-3075, Volume-2, Issue-6,May-2013.

[11]. Premananda B.S, Samarth S. Pai, Shashank B, ShashankS.Bhat, "Design and Implementation of 8-

bit Vedic Multiplier", IJAREEIE, Vol.2, Issue 12, ISSN: 2320-3765, Dec-2013.

[12]. Anju& V.K. Agrawal,"FPGA Implementation of Low Power and High Efficiency Vedic Multiplier using Vedic Mathematics", IOSRJVSP , e-ISSN: 2319 – 4200 ,2, Issue 5 (May. – Jun. 2013), PP 51-57

[13]. Booth, A.D., "A signed binary multiplication technique," Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, pt. 2, pp. 236– 240, 1951

[14]. Jagadguru,Swami Sri Bharath, KrsnaTirathji, "Vedic Mathematics or Sixteen Simple Sutras From The Vedas", MotilalBanarsidas, Varanasi(India),1986

[15]. Mrs. M. Ramalatha, Prof. D. Sridharan, "VLSI Based High Efficiency Karatsuba Multiplier for Cryptographic Applications Using Vedic Mathematics", IJSCI, 2007.

[16]. L. Ciminiera and A. Valenzano, "Low cost serial multipliers for high efficiency specialised processors," Computers and Digital Techniques, IEEE Proc., vol. 135.5, 1988, pp. 259-265.