International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Energy Aware Multiplier Accumulation Unit for Error Tolerant Digital Signal Processing Application

Sumant Mukherjee¹, Saurabh Mitra²

¹Dr. C.V.Raman University, Department of Engineering Kota, Bilaspur,(C.G.) *sjsumant@gmail.com*

²Dr. C.V.Raman University, Department of engineering Kota, Bilaspur,(C.G.) *saurabh.mit1000@gmail.com*

Abstract: In this paper we are trying to implement a Multiplier and accumulation unit which will make justice with SPAA (Speed, Power, Area and Accuracy) metrics. Many authors develop imprecise but simplified MAC units, which provide an extra layer of power savings over conventional low-power design techniques. Due to the limitations in these methods and the accuracy level requirements still the complexity can be reduced and the SPAA metrics can be still achieved efficiently. To improve SPAA metrics still there is no any architecture, so we need a novel MAC unit with low power, high speed, with increased density.

Keywords: Adder, Multiplier, MAC unit, Approximation.

1. Introduction

Digital Signal Processing (DSP) is finding its way into more applications [10], and its popularity has materialized into a number of commercial processors [13]. Digital signal processors have different architectures and features than general purpose processors, and the performance gains of these features largely determine the performance of the whole processor. The demand for these special features stems from algorithms that require intensive computation, and the hardware is often designed to map to these algorithms. Widely used DSP algorithms include the Finite Impulse Response (FIR) filter, Infinite Impulse Response (IIR) filter, and Fast Fourier Transform (FFT)[12].Efficient computation of these algorithms is a direct result of the efficient design of the underlying hardware. One of the most important hardware structures in a DSP processor is the Multiply Accumulate (MAC) unit. Modern computers may contain a dedicated MAC, consisting of a multiplier implemented in combinational logic followed by an adder and an accumulator register that stores the result. This unit can calculate the running sum of products, which is at the heart of algorithms such as the FIR [11] and FFT [12]. The ability to compute with a fast MAC unit is essential to achieve high performance in many DSP algorithms, and is why there is at least one dedicated MAC unit in all of the modern commercial DSP processors [10]. MAC unit must be very efficient. The conventional high level model of the MAC unit after synthesis is shown in figure 1. The multiplier consists of a partial product multiplier that generates the result in carry-save format and a final carry-propagate adder, as the representations. The final adder in figure 1 accumulates the new product to the sum of the previous clock cycle.



Figure 1: The benchmark MAC unit

The Multipliers have an important effect in designing arithmetic, signal and image processors. Many mandatory functions in such processors make use of multipliers (for example, the basic building blocks in Fast Fourier transforms (FFTs) and multiply accumulate (MAC) are multipliers). The advanced digital processors now have fast bit-parallel multipliers embedded in them. Various methods exist for the reduction in the computation time involved by the multiplier with other factors as trade-offs. High-speed, bit-parallel multiplication can be classified into three types

(a) shift-and-add multipliers that generate partial products sequentially and accumulate. This requires more hardware and is the slowest multiplier. This is basically the array multiplier making use of the classical multiplying technique which consumes more time to perform two subtasks, addition and shifting of the bits and hence consumes 2 to 8 cycles of clock period.

(b) Generating all the partial product bits in parallel and accumulate them using a multi-operand adder. This is also called as parallel multiplier by using the techniques of Wallace tree [10] and Booth algorithm [13],

International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

(c) Using arrays of almost identical cells for generation of bit products and accumulation. The uses of Vedic Mathematics shows its application in fast calculations (multiplication, division, squaring, cubing, square root, cube root), trigonometry, three dimensional coordinate geometry, solution of plane and spherical triangles, linear and nonlinear differential equations, matrices and determinants, log and exponential[10]. The most interesting point is to note that the Vedic Mathematics provides unique solutions in several instances where trial and error method is available at present. Vedic Mathematics offers a fresh and highly efficient approach to mathematics covering a wide range starts with elementary multiplication and concludes with a relatively advanced topic, the solution of non-linear partial differential equations. But the Vedic scheme is not simply a collection of rapid methods; it is a system, a unified approach.

It is assumed that a usable circuit/system should function perfectly in conventional digital VLSI design, to always provide definite and accurate results. However, in our nondigital worldly requests, such ideal operations are seldom needed. "Analog computation" that yields "good enough" results instead of totally accurate results [7] may in fact be acceptable. In fact, for many digital systems, the data they process have already contained errors. In applications such as a communication system, the analog signal coming from outside world must first be sampled before we can convert it to digital data at the front end of the system. The digital data is then processed and transmitted in a noisy channel before being converted back to the analog signal at the back end. During this process, errors may occur everywhere. Furthermore, due to the advances in transistor size scaling, the previously insignificant factors such as noise and process variations are becoming important impacts in today's digital IC design [13].

To grapple with aforementioned VLSI design dilemmas, one central and novel approach: Imprecise/approximate design have been anticipated by International Technology Roadmap for Semiconductors (ITRS) [10]. The expediency of imprecise/ approximate designs can be taken blessing of a novel SPAA metrics, resulting astonishing improvements in speed and/or power with a feeble accord in accuracy. Moreover, in recent years, emerging class of killer applications that manifest inherent error-resilience such as multimedia, graphics, and wireless communications has enraptured the concept of imprecise/approximate design that became a state-of-the-art demand. For example, in most multimedia applications, DSP blocks are implemented as their core that process signals relevant to human senses, e.g., sight and hearing. The verity of limited perception of human senses alleviates the constraints on accuracy, showing amity with imprecise/approximate designs.

In accordance with error-resilience design, conventional wisdom investigate several mechanisms such as truncation [1] [2], over-clocking, and voltage over-scaling (VOS) [3], which were not able to configure SPAA metrics effectively. Beyond this end, some other scenarios such as functional approximations have been advised in literatures. Conventional adders such as carry-select (CSL) and carry-look-ahead (CLA) have critical-path delay approximately logN, where N is bit-width; on the other hand,

imprecise/approximate adders have been proposed by researchers via the concept of shorten carry-chain to elevate design performance. As imprecise/approximate design has attracted significant interest in recent years, apart the different design techniques, researchers investigate modeling, analysis, and synthesis approach for imprecise/approximate designs.

This work proposes a Approximate multiplier providing the solution of the aforesaid problems adopting the sutra of Vedic Mathematics called Urdhva Tiryakbhyam (Vertically and Cross wise)[1,2,3]. It can be shown that the design MAC unit is highly efficient in terms silicon area/speed.

2. Literature Review

2.1 Adder Algorithms and Implementations

In nearly all digital IC designs today, the addition operation is one of the most essential and frequent operations. Instruction sets for DSP's and general purpose processors include at least one type of addition. Other instructions such as subtraction and multiplication employ addition in their operations, and their underlying hardware is similar if not identical to addition hardware. Often, an adder or multiple adders will be in the critical path of the design, hence the performance of a design will be often be limited by the performance of its adders. When looking at other attributes of a chip, such as area or power, the designer will find that the hardware for addition will be a large contributor to these areas. It is therefore beneficial to choose the correct adder to implement in a design because of the many factors it affects in the overall chip. In this part we begin with the basic building blocks used for addition, then go through different algorithms and name their advantages and disadvantages.

3. Basic Adder blocks

3.1 Half Adder

The Half Adder (HA) is the most basic adder. It takes in two bits of the same weight, and creates a sum and a carryout. The equation (1) and (2) are the Boolean equations for sum and carryout, respectively.

sum = a xor b (1) carryout = a and b (2)

3.2 Full Adder

The Full Adder (FA) is useful for additions that have multiple bits in each of its operands. It takes in three inputs and creates two outputs, a sum and a carryout. Equation (3), (4) and (5) are the Boolean equations for the FA sum and FA carryout, respectively. In both those equations cin means carrying.

sum = a xor b xor cin(3)

carryout = a and b + b and cin + a and cin (4)

cin = a and b + (a + b)and cin(5)

From the above equations we see that sum and carryout is depends on carryin.

3.3 Partial Full Adder

The Partial Full Adder (PFA) is a structure that implements intermediate signals that can be used in the calculation of the carry bit. Revisiting the truth table for a FA, we extend it to include the signals generate (g), delete (d), and propagate (p). When g=1, it means carryout will be 1 (generated) regardless of carryin. When d=1, it means carryout will be 0 (deleted) regardless of carryin. When p=1, it means carryout will equal carryin (carryin will be propagated). The Boolean equations for the sum and carryout can now be written as functions of g,p, or d. Equation, p must be implemented. with the XOR). Figure 3 shows a circuit for creating the generate, propagate, and sum signals. It is a partial full adder because it does not calculate the carryout signal directly, rather it creates the signals needed to calculate the carryout signal..



Figure 2: Gate Schematic for a Partial Full Adder (PFA) Table 1: Extended Truth Table for a 1-bit adder

| Inputs | | | Outputs | | | | | |
|---------|---|---|----------|-----|---|---|---|--------------------|
| carryin | a | b | carryout | sum | g | d | p | Carry status |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | delete |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | propagate |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | propagate |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | generate/propagate |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | delete |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | propagate |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | propagate |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | generate/propagate |

generate(g) = a and b (6) delete(d) = \overline{a} and \overline{b} (7) propagate(p) = a and b (or a xor b) (8) sum = p xor carryin (9) carryout = g or p and carryin (10)

3.4 Ripple Carry Adder [14]

The Ripple Carry Adder (RCA) is one of the simplest adders to implement. This adder takes in two N-bit inputs (where N is a positive integer) and produces (N + 1) output bits (an Nbit sum and a 1-bit carryout). The RCA is built from N full adders cascaded together, with the carryout bit of one FA tied to the carryin bit of the next FA.The input operands are labeled a and b, the carryout of each FA is labeled cout (which is equivalent to the carryin (cin) of the subsequent FA), and the sum bits are labeled sum. Each sum bit requires both input operands and cin before it can be calculated. To estimate the propagation delay of this adder, we should look at the worst case delay over every possible combination of inputs. This is also known as the critical path. The most significant sum bit can only be calculated when the carryout of the previous FA is known. In the worst case (when all the carry outs are 1), this carry bit needs to ripple across the structure from the least significant position to the most significant position. Hence, the time for this implementation of the adder is expressed in Equation (11), where tRCAcarry is the delay for the carryout of a FA and tRCAsum is the delay for the sum of a FA.

Propagation Delay(tRCAprop) = (N - 1). tRCAcarry + tRCAsum (11)

From Equation (11) , we can see that the delay is proportional to the length of the adder. An example of a worst case propagation delay input pattern for a 4 bit ripple carry adder is where the input operands change from 1111 and 0000 to 1111 and 0001, resulting in a sum changing from 01111 to 10000. From a VLSI design perspective, this is the easiest adder to implement. One just needs to design and layout one FA cell, and then array N of these cells to create an N-bit RCA. The performance of the one FA cell will largely determine the speed of the whole RCA. From the critical path in Equation , minimizing the carryout delay (tRCAcarry) of the FA will minimize tRCAprop . There are various implementations of the FA cell to minimize the carryout delay [2].

3.5 Carry Look Ahead Adder [15]

From the critical path equations in above Sections and 2th delay is linearly dependent on N, the length of the adder. It is also shown in Equations and that the t carryout signal contributes largely to the delay. An algorithm that reduces the time to calculate t carryout and the linear dependency on N can greatly speed up the addition operation. Equation shows that the carryout can be calculated with g, p, and carry in. The signals g and p are not dependent on carry in, and can be calculated as soon as the two input operands arrive. Weinberger and Smith invented the Carry Look Ahead (CLA) Adder [5]. Each subsequent carryout generated becomes increasingly difficult because of the large number of high fan-in gates [6]. We see that the delay for a CLA adder is dependent on the number of levels of carry logic, and not on the length of the adder. If a group size of four is chosen, then the number of levels in an N-bit CLA and in general the number of levels in a CLA for a group size of k. For an N-bit CLA adder, each level of carry logic introduces two gate delays in addition to a gate delay for generate and propagate signals and a gate delay for the sum. This theoretically results in one of the fastest adder architectures.

4. Multiplication Schemes

Multiplication hardware often consumes much time and area compared to other arithmetic operations. Digital signal processors use a multiplier/MAC unit as a basic building block [5] and the algorithms they run are often multiplyintensive. A multiplication operation can be broken down into two steps:

1) Generate the partial products.

2) Accumulate (add) the partial products.



Figure 3: Generic Multiplier Block Diagram



Figure 4: Partial product array for an M *N multiplier: As shown it implies that M = N

4.1 Array Multiplier

From Equation, each multiplicand is multiplied by a bit in the multiplier, generating N partial products. Each of these partial products is either the multiplicand shifted by some amount, or 0. This is illustrated in Fig for an M * N multiplies operation. The generation of partial products consists of simple AND'ing of the multiplier and the multiplicand. The accumulation of these partial products can be done with rows of ripple adders. Thus, the carry out from the least significant bit ripples to the most significant bit of the same row, and then down the left side" of the structure. The partial products are added in ripple fashion with half and full adders. A full adder's inputs require the carry in from the adjacent full adder in its row and the sum from a full adder in the above row. Abdelgawad [9] states that finding the critical path in this structure is non-trivial, but once identified, results in multiple critical paths. It requires a lot of time to optimize the adders in the array since all adders in the multiple critical paths need to be optimized to result in any speed increase (this implies optimization of both the sum and carryout signals in a full adder). The delay basically comes down to a ripple delay through a row, and then down a column, so it is linearly proportional (td *(M +N)) to the sum of the sizes of the input operands.

4.2 Tree Multiplier

The tree multiplier reduces the time for the accumulation of partial products by adding all of them in parallel, whereas the array multiplier adds each partial product in series. The tree multiplier commonly uses CSAs to accumulate the partial products.

4.2.1 Wallace Tree

The reduction of partial products using full adders as carrysave adders (also called 3:2 counters) became generally known as the \Wallace Tree" [14]. Figure shows an example of tree reduction for an 8*8-bit partial product tree. The ovals around the dots represent either a full adder (for three circled dots) or a half adder (for two circled dots). This tree is reduced to two rows for a carry-propagate adder after four stages. There are many ways to reduce this tree with CSAs, and this fig is just one of them.



Figure 5: Wallace Tree for an 8 * 8-bit partial product tree

4.3 Vedic Multiplication

4.3.1 History of Vedic Multiplication

Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya- Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda. It covers explanation of several modern mathematical terms including arithmetic, geometry (plane, co-ordinate), trigonometry, quadratic equations, factorization and even calculus. Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc.

4.3.2 Algorithms of Vedic Mathematics

The proposed Vedic multiplier is based on the Vedic multiplication formulae (Sutras). These Sutras have been traditionally used for the multiplication of two numbers in the decimal number system. In this work, we apply the same ideas to the binary number system to make the proposed algorithm compatible with the digital hardware. Vedic multiplication based on some algorithms, some are discussed below:

4.3.3 Urdhva- Triyagbhyam (Vertically & Crosswise)

Urdhva tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and Crosswise". The conventional methods already know to us will require 16 multiplications and 15 additions. Urdhva-tiryakbyham[8] (Vertically and crosswise) deals with the Multiplication of numbers. The sutra has been traditionally used for the Multiplication of decimal number [10]. These are having several steps to be followed:-let us assume 2 digits no, 46 and 55.

- Multiply the unit place digit, store its value in unit place while carry generated is forwarded to next no.(place value).
- Now perform the Cross Multiplication of Unit and Ten's place value and store the result in addition with previous carry and multiplication value. If carry generated forward it to next place value.

• Now next perform the multiplication to the tens place digit and then add the result with forwarded carry.



Algorithm for 4 x 4 bit Vedic multiplier Using Urdhva Tiryakbhyam (Vertically and crosswise) for two Binary numbers

X3 X2 X1 X0 (Multiplicand) Y3 Y2 Y1 Y0 (Multiplier)

P6 P5 P4 P3 P2 P1 P0 (Product)

.....

Parallel Computation Methodology(cross product) $X0/YO = XO \times Y0 = P0$ $X1X0 / Y1Y0 = (X1 \times Y0) + (X0 \times Y1) = P1$ X2X1X0 / Y2Y1Y0 = $(X2 \times Y0) + (X0 \times Y2) + (X1 \times Y1) = P2$ X3X2X1X0 / Y3Y2Y1Y0 = $(X3 \times Y0) + (X0 \times Y3) + (X2 \times Y1) + (X1 \times Y2) = P3$ X3X2X1 / Y3Y2Y1 = $(X3 \times Y1) + (X1 \times Y3) + (X2 \times Y2) = P4$ $X3X2 / Y3Y2 = X3 \times Y2) + (X2 \times Y3) = P5$ $X3 / Y3 = X3 \times Y3 = P6$

5. Conclusion

This paper present design and implement a MAC [4, 5, 7] unit . In this work, our main focus is on performance and accuracy, but we do provide some numbers for the arithmetic units relating to energy and power. This is to provide an estimate of the amount of energy and power consumed by the units we choose to implement. Typically, embedded computing systems are required to achieve a required level of computing performance, with simultaneous and severe constraints on their characteristic such as power consumption, mobility and size. Moore's law and the associated shrinking of transistor sizes, increase in mobility, decrease in size and power consumption has served as a driver for the proliferation and ubiquity of embedded systems. It is desirable for this trend to continue, to enable new applications and novel contexts in which embedded systems could be used. So due to this we use approximation, which will minimize size of chip.

References

[1] Leem, L.; Hyungmin Cho; Bau, J.; Jacobson, Q.A.; Mitra, S, "ERSA: Error Resilient System Architecture for probabilistic applications," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010* , vol., no., pp.1560,1565, 8-12 March 2010

- [2] Ning Zhu; Wang-Ling Goh; Kiat-Seng Yeo, "An enhanced low-power high-speed Adder For Error-Tolerant application," *Integrated Circuits, ISIC '09. Proceedings of the 2009 12th International Symposium on*, vol., no., pp.69,72, 14-16 Dec. 2009
- [3] Kahng, A.B.; Seokhyeong Kang, "Accuracyconfigurable adder for approximate arithmetic designs," *Design Automation Conference (DAC)*, 2012 49th ACM/EDAC/IEEE, vol., no., pp.820,825, 3-7 June 2012
- [4] Rudagi, J M; Ambli, Vishwanath; Munavalli, Vishwanath; Patil, Ravindra; Sajjan, Vinaykumar, "Design and implementation of efficient multiplier using Vedic Mathematics," Advances in Recent Technologies in Communication and Computing (ARTCom 2011), 3rd International Conference on , vol., no., pp.162,166, 14-15 Nov. 2011
- [5] Abdelgawad, A.; Bayoumi, M., "High Speed and Area-Efficient Multiply Accumulate (MAC) Unit for Digital Signal Prossing Applications," *Circuits and Systems*, 2007. ISCAS 2007. IEEE International Symposium on , vol., no., pp.3199,3202, 27-30 May 2007
- [6] Mottaghi-Dastjerdi, M.; Afzali-Kusha, A.; Pedram, M., "BZ-FAD: A Low-Power Low-Area Multiplier Based on Shift-and-Add Architecture," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol.17, no.2, pp.302,306, Feb. 2009
- [7] Tung Thanh Hoang; Sjalander, M.; Larsson-Edefors, P., "A High-Speed, Energy-Efficient Two-Cycle Multiply-Accumulate (MAC) Architecture and Its Application to a Double-Throughput MAC Unit," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol.57, no.12, pp.3073,3081, Dec. 2010
- [8] Lomte, R.K.; Bhaskar, P.C., "High Speed Convolution and Deconvolution Using Urdhva Triyagbhyam," VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on, vol., no., pp.323,324, 4-6 July 2011
- [9] Abdelgawad, A., "Low power multiply accumulate unit (MAC) for future Wireless Sensor Networks," *Sensors Applications Symposium (SAS), 2013 IEEE*, vol., no., pp.129,132, 19-21 Feb. 2013
- [10] Saokar, S.S.; Banakar, R. M.; Siddamal, S., "High speed signed multiplier for Digital Signal Processing applications," *Signal Processing, Computing and Control (ISPCC), 2012 IEEE International Conference* on, vol., no., pp.1,6, 15-17 March 2012
- [11] Gandhi, D.R.; Shah, N.N., "Comparative analysis for hardware circuit architecture of Wallace tree multiplier," *Intelligent Systems and Signal Processing* (ISSP), 2013 International Conference on , vol., no., pp.1,6, 1-2 March 2013
- [12] Prakash, A.R.; Kirubaveni, S., "Performance evaluation of FFT processor using conventional and Vedic algorithm," *Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN), 2013 International Conference on*, vol., no., pp.89,94, 25-26 March 2013
- [13] Itawadiya, A.K.; Mahle, R.; Patel, V.; Kumar, D., "Design a DSP operations using vedic mathematics," *Communications and Signal Processing (ICCSP), 2013 International Conference on*, vol., no., pp.897,902, 3-5 April 2013

- [14] Khan, S.; Kakde, S.; Suryawanshi, Y., "VLSI implementation of reduced complexity wallace multiplier using energy efficient CMOS full adder," *Computational Intelligence and Computing Research* (ICCIC), 2013 IEEE International Conference on , vol., no., pp.1,4, 26-28 Dec. 2013
- [15] Yu-Ting Pai; Yu-Kumg Chen, "The fastest carry lookahead adder," *Field-Programmable Technology*, 2004. Proceedings. 2004 IEEE International Conference on , vol., no., pp.434,436, 28-30 Jan. 2004