SPAA Aware Multiplier Accumulation Unit for Error Tolerant Digital Image Processing Application

Sudhir S. Kanade
Professor and Head Electronics and Telecommunication Engineering Department
TPCT's College Of Engineering, Osmanabad, (M.S.) India.

Padmanabh D. Deshpande
Electronics and Telecommunication Engineering Department
TPCT's College Of Engineering, Osmanabad, (M.S.), India.

Abstract— Digital Image Processing (DIP) is finding its way into more applications, and its popularity has materialized into a number of commercial processors. As we know every Digital Image processing applications like FFT, FIR, DCT, Smooth filter etc are require multiplier and adder unit. These adder and multiplier unit is heart of those algorithms. As we know there are many applications in DSP and DIP which are error tolerant so for those application we can design error resilient Multiplier and accumulation unit make justice with SPAA (Speed, Power, Area and Accuracy) metrics. In this paper we propose energy aware error tolerant MAC unit which has reduce the hardware complexity and make justice with SPAA metrics. We also implement one of most important algorithm 2D Gaussian smooth filter, on these algorithm we apply proposed multiplier and adder unit for analysis of application efficiency. We also use many image quality parameters for analysis of generated image from proposed design. Those parameter are like PSNR (Peak signal to noise ratio), SSIM (Structural similarity image parameter), RFSIM (RIESZ transformed based feature similarity index), FSIM (Feature similarity index) and GMSD (Gradient magnitude standard deviation).We also propose a hardware unit of proposed MAC unit and compare propose design with existing designs.

Keywords— Accuracy, Adder, Area, Multiplier and Accumulation Unit, Power, Speed.

1. Introduction

THE multiply-accumulate (MAC) unit is a common digital block used extensively in microprocessors and digital signal processors for data-intensive applications. For example, many filters, orthogonal frequency-division multiplexing algorithms, and channel estimators require FIR or FFT/IFFT computations that MAC units can accelerate efficiently. Digital Signal Processing (DSP) is finding its way into more applications, and its popularity has materialized into a number of commercial processors. 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).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. This unit can calculate the running sum of products, which is at the heart of algorithms such as the FIR. In Gandhi, D.R and Shah, N.N [17] and FFT in Prakash, A.R and Kirubaveni, S. [18]. 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. In Saokar, S.S., Banakar, R.M. and Siddamal, S. [15]. 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 enerates the result in carry-save format and a final carry-propagate adder, as the converter between the two different number representations. The final adder in figure 1 accumulates the new product to the sum of the previous clock cycle.

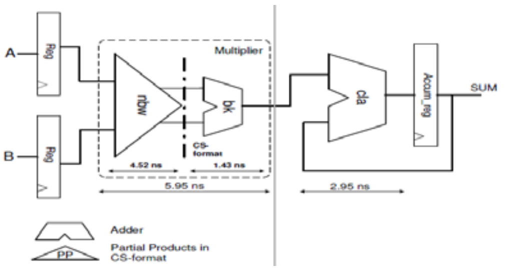


Fig. 1. The benchmark MAC unit

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 in Saokar, S.S., Banakar, R.M. and Siddamal, S. [15] and Booth algorithm in Itawadiya, A.K., Mahle, R., Patel, V. and Kumar, D., [19]

(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 non-linear differential equations, matrices and determinants, log and exponential in Saokar, S.S., Banakar, R.M. and Siddamal, S. [15]. 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 non-digital worldly requests, such ideal operations are seldom needed. "Analog computation" that yields "good enough" results instead of totally accurate results in Tung Thanh Hoang, Sjalander, M. and Larsson-Edefors, P. [11] 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 in Itawadiya, A.K., Mahle, R., Patel, V. and Kumar, D. [19].

Objectives

The goal of this proposed scheme is to design and implement a MAC similar in Rudagi, J.M., Ambli, Vishwanath, Munavalli Vishwanath, Patil, Ravindra and Sajjan, Vinaykumar [7] ,Abdelgawad, A. and Bayoumi, M. [8], Tung Thanh Hoang, Sjalander, M. and Larsson-Edefors, P.; [11] unit . In this paper, 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. The priorities of this objective, in order of importance, are:

1) Robust and safe circuits.
2) Design time
3) Area/speed balance

2. Literature Review

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 be in the critical path of the design, hence the performance of a design often be limited by the performance of its adders. When looking at other attributes of a chip, such as area or power, the designer find that the hardware for addition 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 chapter we begin with the basic building blocks used for addition, then go through different algorithms and name their advantages and disadvantages. We then discuss the design and implementation of the adder chosen for use in a single processor on the AsAP architecture.

2.1 Basic Adder blocks

- 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. Table 1 shows the truth table for this adder. If the two inputs a and b have a weight of 2i (where i is an integer), sum has a weight of 2i, and carryout has a weight of 2(i+1). The equations are the Boolean equations for sum and carryout, respectively.

sum = a xor b
carryout = a and b

TABLE 1
Truth table for a Half Adder

Inputs		Outputs	
a	*b*	*carryout*	*sum*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- 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. The inputs have the same weight, 2i, the sum output has a weight of 2i, and the carryout output has a weight of 2(i+1). The truth table for the FA is shown in Table 2. The FA differs from the HA in that it has a carrying as one of its inputs, allowing for the cascading of this structure which is explored. Equations are the Boolean equations for the FA sum and FA carryout, respectively. In both those equations cin means carrying.

TABLE 2
Truth table for a Full Adder

Inputs			Outputs	
carryin	*a*	*b*	*carryout*	*sum*
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- 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 (Carry-Save Adders) to accumulate the partial products.

2.1.1 Wallace Tree

The reduction of partial products using full adders as carry-save adders (also called 3:2 counters) became generally known as the "Wallace Tree" in Khan, S., Kakde, S. and Suryawanshi, Y. [20]. 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 example is just one of them.

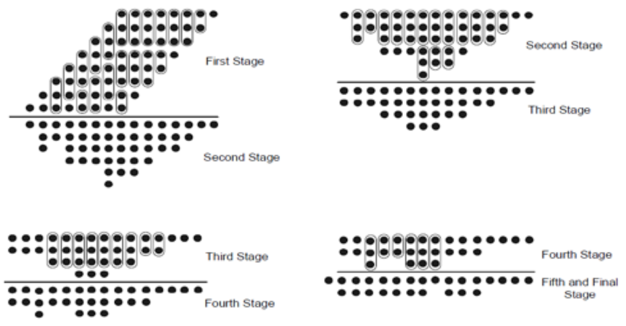


Fig. 2 Wallace Tree for an 8 * 8-bit partial product tree

2.2 VEDIC MULTIPLICATION ALGORITHMS

HISTORY OF VEDIC MATHEMATICS:-

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. His Holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884-1960) comprised all this work together and gave its mathematical explanation while discussing it for various applications. Swahiji constructed 16 sutras (formulae) and 16 Upa sutras (sub formulae) after extensive research in Atharva Veda. Obviously these formulae are not to be found in present text of Atharva Veda because these formulae were constructed by Swamiji himself. Vedic mathematics is not only a mathematical wonder but also it is logical. That is why VM has such a degree of eminence which cannot be disapproved. Due these phenomenal characteristic, VM has already crossed the boundaries of India and has become a leading topic of research abroad. VM deals with several basic as well as complex mathematical operations. Especially, methods of basic arithmetic are extremely simple and powerful in Abdelgawad, A. [14],Saokar, S.S., Banakar, R.M. and Siddamal, S.; [15].

The word „Vedic is derived from the word „Vedic which means the store-house of all knowledge. Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc. These Sutras along with their brief meanings are enlisted below alphabetically

1)(Anurupye) Shunyamanyat – If one is in ratio, the other is zero.
2) Chalana-Kalanabyham – Differences and Similarities.
3)Ekadhikina Purvena – By one more than the previous One.
4)Ekanyunena Purvena – By one less than the previous one.
5)Gunakasamuchyah – The factors of the sum is equal to the sum of the factors.
6)Gunitasamuchyah – The product of the sum is equal to

the sum of the product.
7)Nikhilam Navatashcaramam Dashatah – All from 9 and last from 10.
8) Paraavartya Yojayet – Transpose and adjust.
9)Puranapuranabyham – By the completion or noncompletion.
10)Sankalana- vyavakalanabhyam – By addition and by subtraction.
11)Shesanyankena Charamena – The remainders by the last digit.
12)Shunyam Saamyasamuccaye – When the sum is the same that sum is zero.
13)Sopaantyadvayamantyam – The ultimate and twice the penultimate.
14) Urdhva-tiryakbhyam – Vertically and crosswise.
15) Vyashtisamanstih – Part and Whole.
16)Yaavadunam – Whatever the extent of its deficiency.
These methods and ideas can be directly applied to trigonometry, plain and spherical geometry, conics, calculus (both differential and integral), and applied mathematics of various kinds. As mentioned earlier, all these Sutras were reconstructed from ancient Vedic texts early in the last century. Many Sub-sutras were also discovered at the same time, which are not discussed here. The beauty of Vedic mathematics lies in the fact that it reduces the otherwise cumbersome-looking calculations in conventional mathematics to a very simple one. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing and digital signal processing in Tiwari, H. D. and Gankhuyag, G. [10],Tung Thanh Hoang, Sjalander, M. and Larsson-Edefors, P. [11].

The multiplier architecture can be generally classified into three categories. First is the serial multiplier which emphasizes on hardware and minimum amount of chip area. Second is parallel multiplier (array and tree) which carries out high speed mathematical operations. But the drawback is the relatively larger chip area consumption. Third is serial- parallel multiplier which serves as a good trade-off between the times consuming serial multiplier and the area consuming parallel multipliers.

MAC UNIT

In this paper Leem, L., Jacobson, Q.A. and Mitra, S. [3] authors present a new approach to increase the performance of MAC by using ancient Vedic mathematics principles. As multiplier is in the critical component of MAC author focuses more on reducing the complexity of Multiplier. The proposed multiplier is designed using the principle of Urdhva Triyagbhyam which means vertically and crosswise . The results show an improvement in area and speed of the multiplier performance in comparison to conventional architectures. The drawback of this approach is Multiplier still lies in Critical path of Unit. Unnecessary active power consumption of Multiplier block where end applications does not require. Power consumption is still critical parameter in performance evaluation.

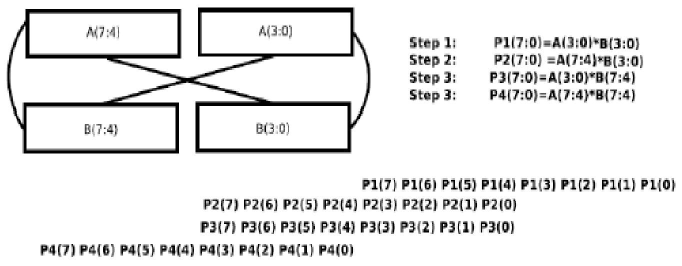

A(7:4)
A(3:0)
B(7:4)
B(3:0)
Step 1: P1(7:0)=A(3:0)*B(3:0)
Step 2: P2(7:0) =A(7:4)*B(3:0)
Step 3: P3(7:0)=A(3:0)*B(7:4)
Step 3: P4(7:0)=A(7:4)*B(7:4)
P1(7) P1(6) P1(5) P1(4) P1(3) P1(2) P1(1) P1(0)
P2(7) P2(6) P2(5) P2(4) P2(3) P2(2) P2(1) P2(0)
P3(7) P3(6) P3(5) P3(4) P3(3) P3(2) P3(1) P3(0)
P4(7) P4(6) P4(5) P4(4) P4(3) P4(2) P4(1) P4(0)

Fig. 3 Multiplication Performance and Evaluation.

There is another MAC unit architecture In paper Abdelgawad, A. and Bayoumi, M. [8] authors presents a high speed and area efficient MAC . Realization of High speed and area efficient in MAC is achieved by reducing the hardware complexity and critical delays or critical path in MAC unit. A new architecture is built using a 4:2 binary tree compressor i.e instead of 3:2 binary tree adder the authors bypassed the input addend directly to the carry propagate block of multiplier. The drawback of this approach is proposed 4:2 compressed binary tree result in long carry propagation which increases delay for long carry chains. In this paper Mottaghi-Dastjerdi, M., Afzali-Kusha, A. and Pedram, M. [9] author presents a new architecture that supports two's complement numbers, and includes accumulation guard bits and saturation circuitry. The first MAC pipeline stage contains only partial-product generation circuitry and a reduction tree, while the second stage. Realization of High speed and area efficient in MAC is achieved by reducing the hardware complexity and critical delays or critical path in MAC unit.

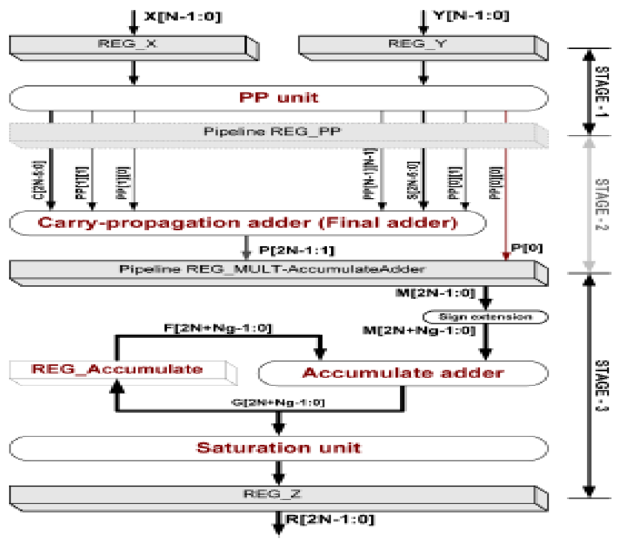


Fig.4 3-Stage MAC Unit

3. Design and Implementation Of MAC Unit For Error Tolerant DIP Application

3.1 Error Resilient System Architecture

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. However, our ability to miniaturize silicon-based transistors is under serious jeopardy. These challenges can broadly be classified under two categories (i) the change in the nature of materials and material properties as the sizes of the transistors decrease. and (ii) our inability to fabricate identical and reliable nanometer-sized silicon devices and achieve uniform behavioral characteristics. These challenges affect the physical characteristics of transistors and hence computing platforms in many ways. For example, devices are no longer expected to behave in a deterministic and reliable manner, and the probabilistic and unreliable behavior of devices isdeemed inevitable by the international technology road-map for semiconductors (itrs) which forecasts "Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification, and test. Such a paradigm shift is likely forced in any case by technology scaling, which leads to more transient and permanent failures of signals, logic values, devices, and interconnects.". This non uniform, probabilistic and unreliable behavior of transistors has an impact on the desirable characteristics of embedded systems. For example, to provide adequate noise immunity, the supply voltage of transistors are not scaled down at a rate concomitant to the reduction of the size of the transistors in Ning Zhu, Wang-Ling Goh and Kiat-Seng Yeo [4]. This results in an increase in power density as size of the transistors decrease without a corresponding decrease in power consumption. Increasing power density results in bulky cooling components thus severely impacting the mobility of embedded computingplatforms. A comprehensive survey of such challenges to nanometer-sized devices and beyond may be found in Ning Zhu, Wang-Ling Goh and Kiat-Seng Yeo [4]. Several approaches have been adopted to address these challenges to Moore's law. These approaches include rigorous test mechanisms, techniques which correct errors incurred by architectural primitives using temporal and spatial redundancy in Krishna V. Palem [2], Kiyoung kim[6], an increase in parallelism without an increase in the frequency of operation of computing devices, research into novel non-silicon materials for computing, including molecular devices , graphene and optoelectronics, and design automation-based approaches to reduce the impact of undesirable effects such as timing variations and noise susceptibility. By contrast, the central theme of our work, which we refer to as probabilistic and approximate design, is to design computing systems using circuit components which are susceptible to perturbations. We use the term "perturbations" to cover a broad range of phenomena which cause circuit elements to behave in an "incorrect" or "non-uniform" manner.

3.2 Proposed Scheme

In this paper we design fast approximate adder unit. Using this adder unit we can design 8 bit Multiplier unit. Combining both adder and multiplier unit and develop one 8 bit MAC unit. We also apply those multiplier and adder unit on one Digital image processing application which is known as 2D Gaussian Smooth filter. We have use this application for analysis point of view, means we check that our proposed unit is useful for that kind of applications or not. We implement algorithm on Matlab. Hardware will implement by using of Verilog Hardware language. This approach compare with previous approximate adder, multiplier and MAC unit.

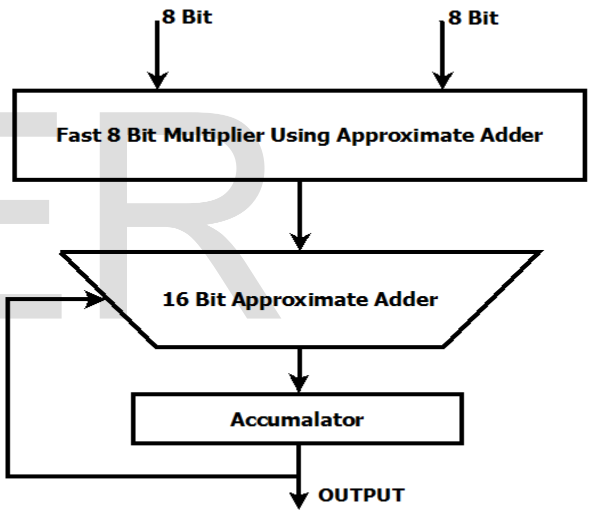


Fig. 5 Block Diagram of MAC Unit

Multiply-Accumulate Unit Design and Implementation

The Multiply-Accumulate (MAC) unit performs the Multiply instruction and the MAC instruction, which are essential for all DSP processors. In order to achieve high performance, the MAC unit is pipelined into three stages. This chapter discusses the design and implementation of the multiply-accumulate unit for the AsAP in Sureka, N. and Porselvi, R.; [16].

The multiplier consists of a partial product multiplier that generates the result and a final carry-look ahead adder, as the converter between the two different number representations. Another method to improve the speed of the multiplication operation is to improve the partial product generation step. This can be done in two ways:

1) Generate the partial products in a faster manner.

2) Reduce the number of partial products that need to be

generated.

$$Z = A \times B + Z$$

Where the multiplier A and multiplicand B are assumed to have n bits each and the addend Z has (2n+1) bits. The basic MAC Unit is made up of a multiplier and an accumulator .The multiplier can also be divided into the partial products generator, summation tree, and final adder. This property leads to four basic blocks to implement. The summation network represents the core of the MAC unit. This block occupies most of the area and consumes most of the circuit power and delay. Several algorithms and architectures are developed in attempt to optimize the implementation of this block.

The addition tree reduces the number of partial products into two operands representing a sum and a carry. The final adder is then used to generate the multiplication result out of these two operands. The last block is the accumulator, which is required to perform a double precision addition operation between the multiplication result and the accumulated operand. This block required a very large adder due to the large operands size. This stage represents a bottleneck in the multiplication process in term of speed since it involves horizontal carry propagation. Tree architectures were proposed to improve the speed of the partial product addition process by introducing parallelism. The tree structure, which was first, introduced using 3-2 compressors suffer from irregular interconnections and is difficult to layout. It also results in high power consumption as a result of the capacitance introduced by large interconnections. A more regular structure is based upon binary trees construction using modified 4-2 compressors.

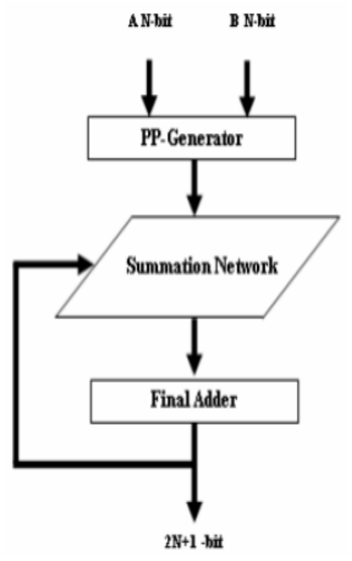


Fig. 6. Summation Network

The merged MAC architecture is based on fully utilizing the summation tree by feeding the accumulated data bits, into the unused inputs of the 4:2 compressors such that the accumulation operation is merged within the multiplication circuit, this will save the cost of an additional accumulator. This can directly result in increasing the overall speed of the MAC operation. Power consumption and circuit area are saved as well. The concept of merged arithmetic was generally introduced in to reduce the implementation cost and improve the processing speed. The architecture is based on t king advantage of the free input lines of the available 4-2 compressors to implement the accumulation operation by feeding the bits of the accumulated operand into the summation tree. The decision of where to insert the bit Z7 determines the number of required modified 4:2 compressors.

Implementation Of 2×2 Bits Vedic Multiplier

It is clear that the basic building blocks of this multiplier are one bit multipliers and adders. One bit multiplication can be performed through two input AND gate and for addition, full adder can be utilized. Lets take two inputs, each of 2 bits; say A1A0 and B1B0. Since output can be of four digits, say Q3Q2Q1Q0. As per basic method of multiplication, result is obtained after getting partial product and doing addition.

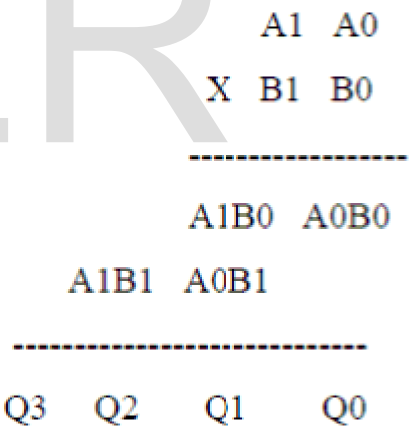


Fig. 7 Implementation Of 2×2 Bits Vedic Multiplier

In Vedic method, Q0 is vertical product of bit A0 and B0, Q1 is addition of crosswise bit multiplication i.e. A1 & B0 and A0 and B1, and Q2 is again vertical product of bits A1 and B1 with the carry generated, if any, from the previous addition during Q1. Q3 outputis nothing but carry generated during Q2 calculation. This module is known as 2x2 multiplier block in Kahng, A. B.;[5]

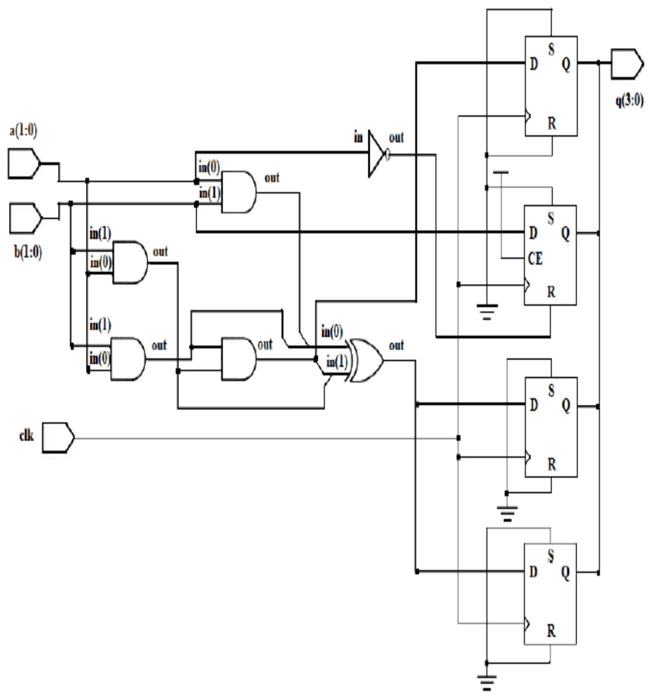


Fig. 8. RTL View of 2×2

Implementation Of 4×4 Bits Vedic Multiplier

Lets analyze 4x4 multiplications, say A3, A2, A1, A0 and B3, B2, B1, B0. Following are the output line for the multiplication result, Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0. Lets divide A and B into two parts, say A3 A2 & A1 A0 for A and B3B2 & B1B0 for B. Using the fundamental of Vedic multiplication, taking two bit at a time and using 2 bit multiplier block,

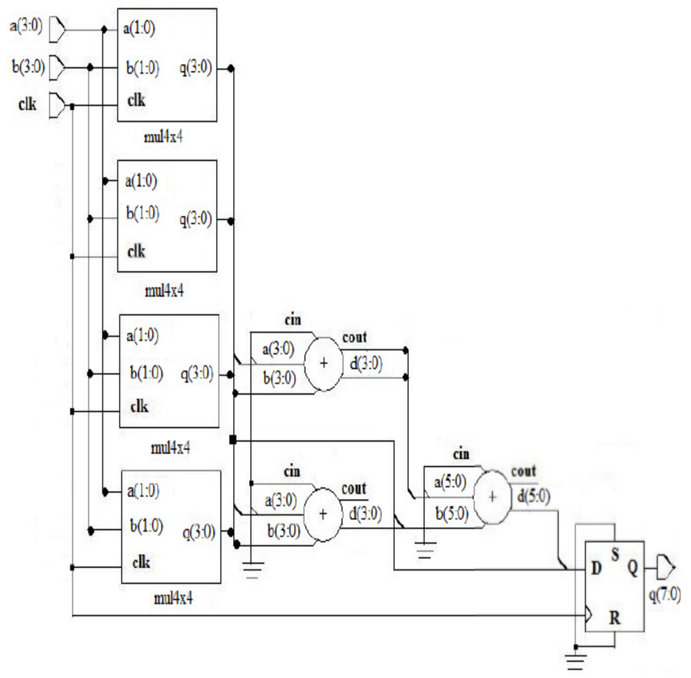
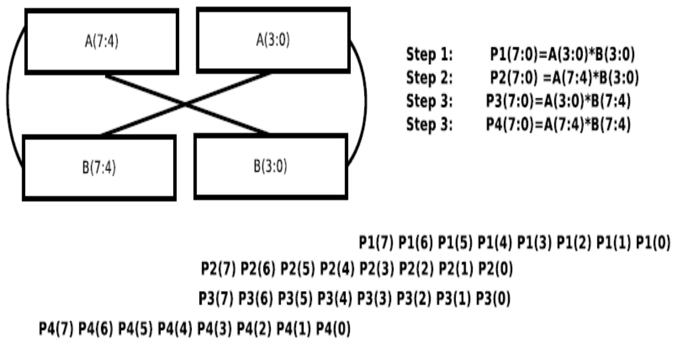


Fig. 9. RTL View of 4×4

Implementation Of 8×8 Bits Vedic Multiplier

The 8x 8 bit multiplier is structured using 4X4 bit blocks as shown in figure . In this figure the 8 bit multiplicand A can be decomposed into pair of 4 bits AH-AL. Similarly multiplicand B can be decomposed into BH-BL. The 16 bit product can be written as:
The 8x 8 bit multiplier is structured using 4X4 bit blocks as


A(7:4)
A(3:0)
B(7:4)
B(3:0)
Step 1: P1(7:0)=A(3:0)*B(3:0)
Step 2: P2(7:0) =A(7:4)*B(3:0)
Step 3: P3(7:0)=A(3:0)*B(7:4)
Step 3: P4(7:0)=A(7:4)*B(7:4)
P1(7) P1(6) P1(5) P1(4) P1(3) P1(2) P1(1) P1(0)
P2(7) P2(6) P2(5) P2(4) P2(3) P2(2) P2(1) P2(0)
P3(7) P3(6) P3(5) P3(4) P3(3) P3(2) P3(1) P3(0)
P4(7) P4(6) P4(5) P4(4) P4(3) P4(2) P4(1) P4(0)

P= A x B= (AH-AL) x (BH-BL)
= AH x BH+AH x BL + AL x BH+ AL x BL
The outputs of 4X4 bit multipli rs are added accordingly to obtain the final product. Thus, in the final stage two adders are also required. Now the basic building block of 8x8 bits Vedic multiplier is 4x4 bits multiplier which implemented in its structural model. For bigger multiplier implementation like 8x8 bits multiplier the 4x4 bits multiplier units has been used as components which are

3.2.1 Partial Product Generation

The first step in a multiplication is to generate the partial product bits. In this implementation, Urdhav multiplication approach is chosen because of the simple task of generating the multiplicand and twice the multiplicand for the partial products. The main advantage of the vedic multiplication algorithm (Urdhva-Tiryak Sutra) stems from the fact that it can be easily realized in hardware. The hardware realization of a 4-bit multiplier using this Sutra is shown in below figure. This hardware design is very similar to that of the famous array multiplier where an array of adders is required to arrive at the final product. All the partial products are calculated in parallel and the delay associated is mainly the time taken by the carry to propagate through the adders which form the multiplication array.

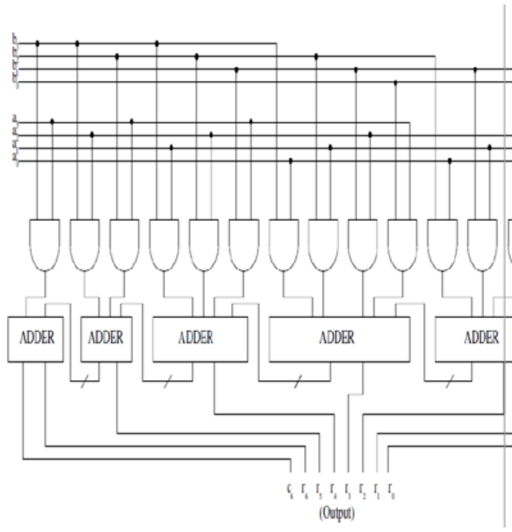


Fig. 10. Bit Multiplier

3.2.2 Accumulation

After the partial products are generated, the next step is to accumulate them. A 8 bit ripple carry is used to accumulate the partial products.

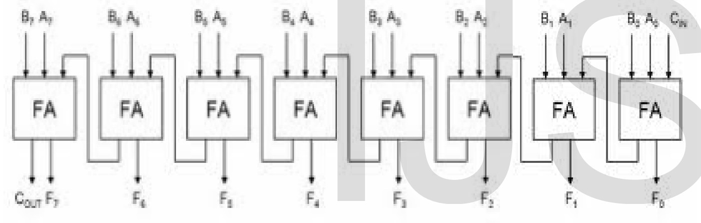


Fig. 11. Accumulation

3.2.3Final Stage for the MAC unit

The final stage of MAC unit is accumulation of the partial products generated by Multiplier. In a conventional adder circuit, the delay is mainly attributed to the carry propagation chain along the critical path, from the Least Significant Bit (LSB) to the Most Significant Bit (MSB). The final adder is then used to generate the multiplication result out of these two operands. The last block is the accumulator, which is required to perform an addition operation between the multiplication result and the accumulated operand. This block required a very large adder due to the large operands size. This stage represents a bottleneck in the multiplication process in term of speed since it involves horizontal carry propagation. Tree architectures were proposed to improve the speed of the partial product addition process by introducing parallelism. The tree structure, which was first, introduced using 3-2 compressors suffer from irregular interconnections and is difficult to layout. It also results in high power consumption as a result of the capacitance introduced by large interconnections. A more regular structure is based upon binary trees construction using modified 4-2 compressors. The below diagram shows the full block of the MAC

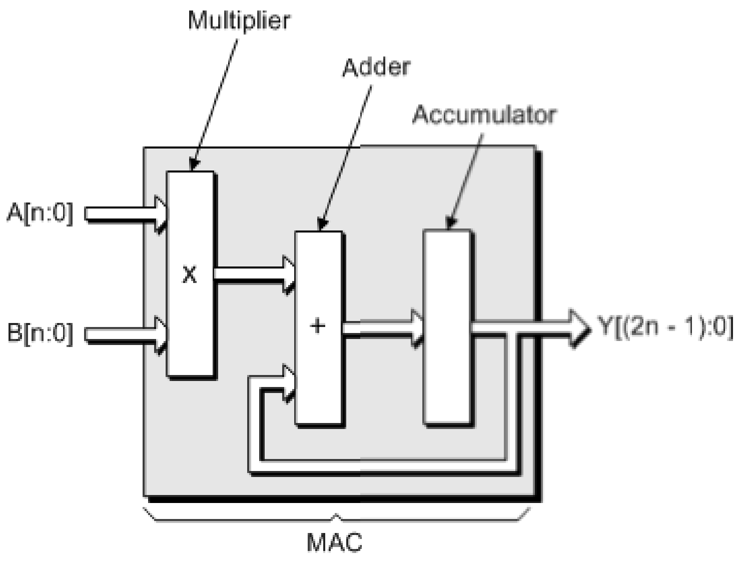


Fig. 12. MAC Unit

4 Result And Discussion

The following section shows the details of the Adders and Multiplier of the techniques discussed in early sections. The implementations of these are done using Verilog HDL in Xilinx tool. The three stages of MAC are implemented using Verilog HDL. Xilinx tool is used to synthesis and simulate the results. The proposed architecture offers a minimum 10%-20% improvement in speed and reduces area 4% to 5%. The first stage of MAC is Multiplier i.e., multiplying the partial products and generating the partial products.

Stage 1: Multiplier

TABLE 3
Multiplier Analysis

	ADD and shift	Wallace Multiplier
Delay	6.002	10.58
LUT's	136	112
Frequency (Mhz)	166	94

From the above comparison we can see that the exisitng multipliers are either larger area consume or having a high delay and high power consumption.

Stage 2: Summation Adder

The second stage of MAC is accumulation of the partial products. The comparison analysis of the adder are done between ripple carry adder and carry look ahead adder.

TABLE 4
Summation Analysis

	Ripplecarry adder	Caary look ahead adder
LUT's	24	17
IOB's	48	48
Delay	4.594	4.316
Frequency	217.67	229.3

Stage 3: Final Accumulation

The last stage of MAC unit is Accumulation and adding the next stage of partial products.

TABLE 4
Accumulation Analysis

	MAC (8)
LUT's	156
Delay (ns)	13.744
Frequency Mhz	72

5. Conclusion

This work explores and analyzes fast adder algorithms and multiplication schemes and utilizes them in the design by implementing MAC unit. From the analysis of the results we can see the usage of MAC is not 100%. So considering this point as the stepping stone to next work, by considering accuracy as one of the parameter the SPAA characterizes can be in cost effective with achieving energy as one of the parameter. The key contribution of this work is to develop a SPAA aware error tolerant MAC Unit. This proposed MAC unit require less area, power and speed. In this approach we propose a new approach of approximation which reduces some amount of accuracy. In this work we propose Approximate Adder unit, using that adder unit we design Multiplier unit, with combination of propose adder and multiplier unit design Novel MAC unit. For application use 2D Gaussian smooth filter and check the quality of generated output from this MAC unit.

References

[1] Garcia, O.N.; Glass, H.; Haimes, S.C., "An approximate and empirical study of the distribution of adder inputs and maximum carry length propagation," Computer Arithmetic (ARITH), 1978 IEEE 4th Symposium on , vol., no., pp.97,103, 25-27 Oct. 1978]

[2] Krishna V. Palem , Lakshmi N.B. Chakrapani , Zvi M. Kedem , Avinash Lingamneni , Kirthi Krishna Muntimadugu, Sustaining moore's law in embedded computing through probabilistic and approximate design: retrospects and prospects, Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems, October 11-16, 2009, Grenoble, France

[3] 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

[4] 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

[5] Kahng, A.B.; Seokhyeong Kang, "Accuracy-configurable adder for approximate arithmetic designs," Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE , vol., no., pp.820,825, 3-7 June 2012

[6] Kiyoung Kim; Taewhan Kim, "Algorithm for synthesizing design context-aware fast carry-skip adders," Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific , vol., no., pp.795,800, Jan. 30 2012-Feb. 2 2012

[7] 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

[8] 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

[9] 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

[10] Tiwari, H.D.; Gankhuyag, G.; Chan-Mo Kim; Yong Beom Cho, "Multiplier design based on ancient Indian Vedic Mathematics," SoC Design Conference, 2008. ISOCC '08. International , vol.02, no., pp.II-65,II-68, 24-25 Nov. 2008

[11] 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

[12] 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

[13] Gupta, A.; Malviya, U.; Kapse, V., "Design of speed, energy and

power efficient reversible logic based vedic ALU for digital processors," Engineering (NUiCONE), 2012 Nirma University International Conference on , vol., no., pp.1,6, 6-8 Dec. 2012

[14] 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

[15] 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

[16] Sureka, N.; Porselvi, R.; Kumuthapriya, K., "An efficient high speed Wallace tree multiplier," Information Communication and Embedded Systems (ICICES), 2013 International Conference on , vol., no., pp.1023,1026, 21-22 Feb. 2013

[17] 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

[18] 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

[19] 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

[20] 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

[21] Baran, D.; Aktan, M.; Oklobdzija, V.G., "Multiplier structures for low power applications in deep-CMOS," Circuits and Systems (ISCAS), 2011 IEEE International Symposium on , vol., no., pp.1061,1064, 15-18 May 2011

[22] 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