FPGA Implementation & Performance Comparision of Various High Speed unsigned Binary Multipliers using VHDL

V.Satya kishore*, J.E.N.Abhilash and G.N.V.Ratnakishor
Deaprtment of Electronics and Communication Engineering,
Swarnandhra College of Engineering and Technology, Seetharampuram, Narsapur, (A.P.), INDIA
*Email: mouniragesh08@gmail.com

***Abstract*—** Today, most of the DSP computations involve the use of multiply accumulate operations and therefore the design of fast and efficient multipliers is imperative. The addition and multiplication of two binary numbers is the fundamental and most often used arithmetic operation in microprocessors, digital signal processors and data-processing application-specific integrated circuits. In this paper, we present the study of different types of multipliers by comparing the speed and area of each. In this work, VHDL coding and XILINX ISE Simulator is employed to implement multipliers like WTM, Dadda Multiplier, Vedic Multiplier, CSHM, Serial Multiplier and Multipliers using different compressors in Wallace tree architecture. The analysis of this work would be helpful to choose a better multiplier in order to fabricate an efficient system.

***Index Terms*—** *ALU; DSP; Wallace tree; Dadda; Vedic Multiplier; CSHM and Compressors.*

*****

I. INTRODUCTION

Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest clement in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, a whole spectrum of multipliers with different area-speed constraints have been designed with fully parallel Multipliers at one end of the spectrum and fully serial multipliers at the other end [1-16]. In between are digit serial multipliers where single digits consisting of several bits are operated on. These multipliers have moderate performance in both speed and area. Multiplication is achieved by adding a list of shifted multiplicands according to the digits of the multiplier.

In this paper, first we have written VHDL code for different multipliers which includes sequential multiplier, Vedic multiplier (Urdhva triyagbhyam & Karustubha algorithms), Computational Sharing Multiplier and Wallace tree multipliers with 3:2/2:2, 4:2 and 5:2 compressors.

II. THEORY & DESIGN APPROACH

A. *Vedic Multiplier Using Urdhya Triyagbhyam*

One of the important Sutras in the Vedic mathematics is Urdhya Triyagbhyam (Vertically and crosswise), deals with the multiplication of numbers [3-6]. To illustrate this multiplication scheme in binary number system, let us consider the multiplication of two binary numbers a3 a2 a1 a0 and b3 b2 b1 b0. As the result of this multiplication would be more than 4 bits, we express it as ...r3 r2 r1 r0. For the sake of in binary system in the last case; the digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result (rn) and a carry (say cn). This carry is added in the next step and hence the process goes on. If more than one line are there in one step, all the results are added to the previous carry. In each step, least significant bit acts as the result bit and the other entire bits act as carry. For example, if in some intermediate step, we get 110, then 0 will act as result bit and 11 as the carry (referred to as cn in this text). It should be clearly noted that cn may be a multi-bit number.

B. *Vedic Multiplier Using Karastuba*

The design of this multiplier starts with Multiplier design that is 2x2 bit multiplier [7]. Here, "Urdhva triyagbhyam Sutra" or "Vertically and Crosswise Algorithm" for multiplication has been effectively used to develop digital multiplier architecture. This algorithm is quite different from the traditional method of multiplication that is to add and shift the partial products. This Sutra shows how to handle multiplication of a larger number (N x N, of N bits each) by breaking it into smaller numbers of size (N/2 = n, say) and these smaller numbers can again be broken into smaller numbers (n/2 each) till we reach multiplicand size of (2 x 2). For Multiplier, first the basic blocks that are the 2x2 bit multipliers can be made and then using these blocks 4x4 block can be implemented. Further, using 4x4 blocks, finally 8x8 bit block can be designed.

C. *Computation Sharing Multiplier*

Computation sharing multiplier used natural binary (NB) numbers for implementation of FIR filter [8]. The computation sharing multiplier (CSHM) architecture and implementation,

ISSN: 2321-8169
3936 - 3940

which is based on the algorithm is presented. CSHM consists of precomputer, select units and adders (S&A). The precomputer produces the multiplication of alphabets with input x and the S&A performs the add and shift operations required to obtain the final output. Figure 1 shows A block diagram of CSHM structure.

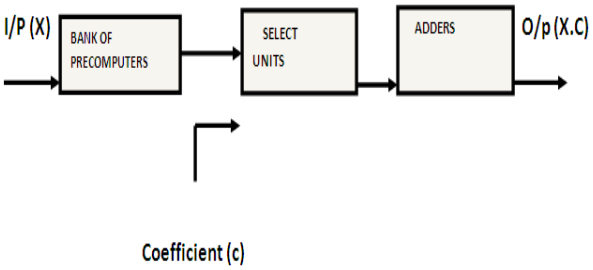

Figure 1. CSHM Structure.

The main advantage of CSHM is that the outputs of pre-computer are shared by all the select units. Hence, the operations performed by select units can be executed in parallel without introducing additional select unit delay. In this design a 8X8 computation sharing multiplier structure can be implemented.

D. Wallace Tree Multiplier

Here we have presented Wallace tree multipliers using 3:2/2:2, 4:2 and 5:2 compressors.

Wallace tree multiplier using 3:2/2:2 compressors: The Wallace tree multiplier belongs to a family of multipliers called column compression multipliers [6]. The principle in this family of multipliers is to achieve partial product accumulated by successively reducing the number of bits of information in each column using full adders or half adders. The full adder is known as (3:2) compressor because of its ability to add three bits from a single column of the partial product matrix and output two bits, one bit in the same column and one bit in the next column of the output matrix. The half adder is known as (2:2) compressor because of its ability to take two bits from a single column of the partial product matrix and output two bits, one bit in the next column of the output matrix.

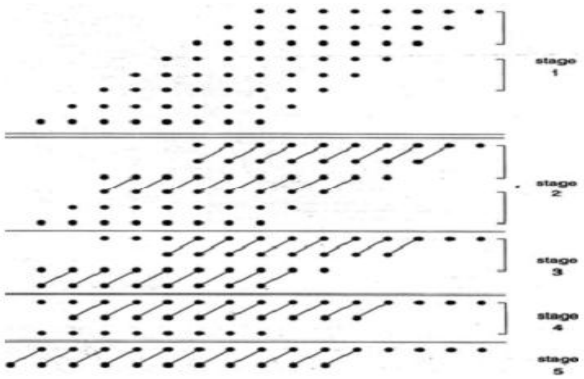

Figure 2. Wallace tree internal structure.

Figure 2 shows dot diagram of Wallace tree multiplier. The Wallace tree consists of numerous levels/ stages of such column compressor structures until finally only two full width operands remain. These two operands can then be added using regular 2N bits adders to obtain the product result.

Wallace tree multiplier using 4:2 and 5:2 compressors: Compressors are arithmetic components, similar in principle to parallel counters but with two distinct differences: they have explicit carry-in and carry-out bits; and there may be some redundancy among the ranks of the sum and carry-output bits. A 4:2 compressor has 4 input bits and produces 2 sum output bits (out0 and out1), a carry-in (cin) and a carry-out (cout) bit (thus, the total number of input/output bits are 5 and 3) as shown in figure 3.

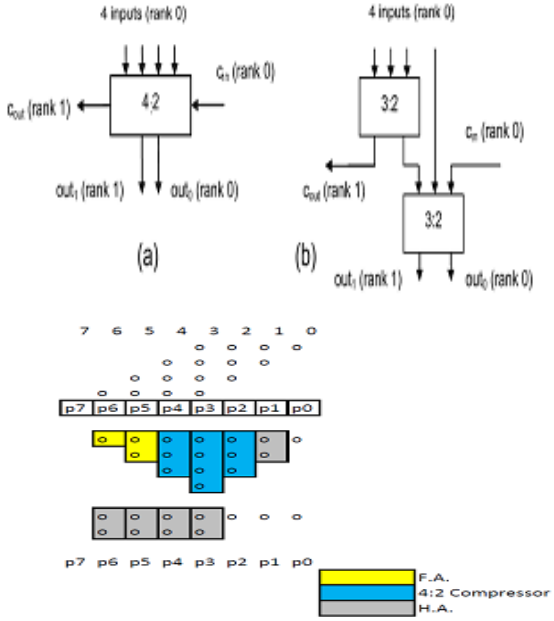

Figure 3. Wallace tree multiplier using 4:2 compressors.

All input bits, including cin, have rank 0; the two output bits have ranks 0 and 1 respectively, while cout has rank 1 as well. Thus, the output of the 4:2 compressor is a redundant number; for example, out1 = 0 and cout = 1 is equivalent to out1 = 1 and cout = 0 in all cases.

A 5:2 compressor has 5 input bits and produces 2 sum output bits (sum and cout3), a carry-in (cin1, cin2) and a carry-out (cout1, cout2,cout3) bit (thus, the total number of input/output bits are 7 and 4) as depicted in figure 4.

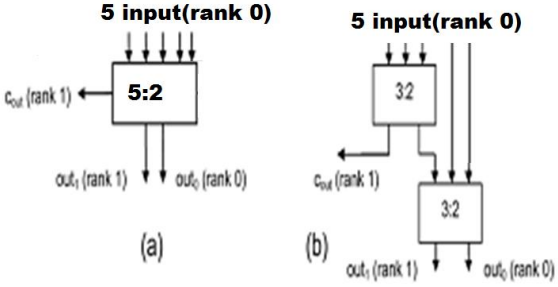

Figure 4. Wallace tree multiplier using 5:2 compressors

ISSN: 2321-8169
3936 - 3940

All input bits, including cin1, have rank 0 and cin2 has rank 2 ; the two output bits have ranks 0 and 1 respectively, while cout2 has rank 1 and cout1 has rank 2 .

E. Dadda Multiplier

Dadda refined Wallace's method by defining a counter placement strategy that required fewer counters in the partial product reduction stage at the cost of a larger carry propagate adder. Dadda has introduced a number of ways to compress the partial product bits using such a counter which later became known as Dadda's Counter. Figure 5 shows the process for 8 × 8 bits Dadda multiplier. An input 8 × 8 bits matrix of dots (each dot represents a bit) is shown as step 0.

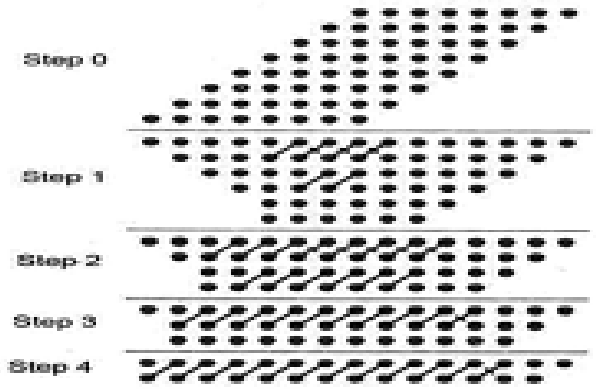

Figure 5. 8X8 Dadda Multiplier architecture.

Columns having more than six dots are reduced by the use of half adders, each half adder takes in two dots and outputs one in the same column and one in the next more significant column and full adders, each full adder takes in three dots and outputs one in the same column and one in the next more significant column so that no column in step 1 will have more than six dots. Half adders are shown by a crossed line in the succeeding matrix and full adders are shown by a line in the succeeding matrix. In each case the rightmost dot of the pair that is connected by a line is in the column from which the inputs were taken from the adder. In the succeeding steps reduction to step 2 with no more than four dots per column, matrix three with no more than three dots per column, and finally step 4 with no more than two dots per column is performed. The height of the matrices is determined by working back from the final two row matrix and limiting the height of the each matrix to the largest integer that is no more than 1.5 times the height of its successor. Each matrix is produced from its predecessor in one adder delay. Since the number of bits in the words to be multiplied, the delay of the matrix reduction process that reduces is proportional to log n, where n is word size. Since the adder that reduces the final two row matrix can be implemented as a CLA which also has logarithmic delay, the total delay for this multiplier is proportional to the logarithm of the word size n.

III. RESULTS & DISCUSSION

All the designed multipliers were synthesized using FPGA kit and results are discussed here. All the designed multipliers have shown same simulation results however, here we have discussed the result of Dadda multiplier as shown in figure 6. This multiplier uses different 8 bit inputs and gives 16 bit results. Our obtained results showed the validity with others reported works which are mainly concentrated on area, delay and power consumption.

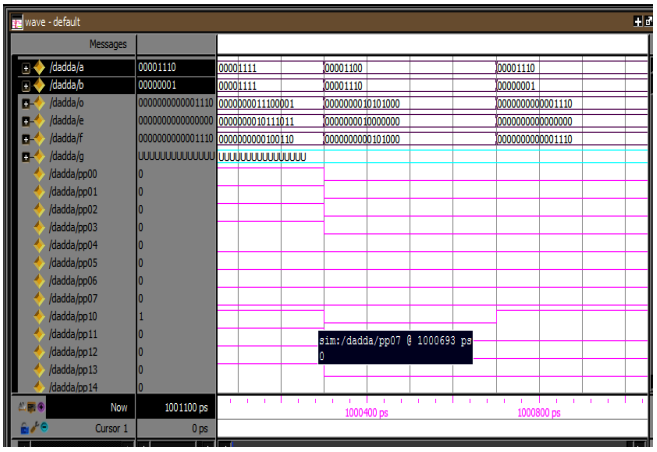

Figure 6. Simulation output waveform of Dadda.

Figure 7 reveals that the sequential serial Add and Shift multiplier requires more number of slices, LUT's and high delay because it takes 8 clock cycles for getting the result as it will perform each and every bit calculation individually that means 8 bits requires 8 clock cycles. Remaining all multipliers are combinational multipliers and it will take less time and less power consumption.

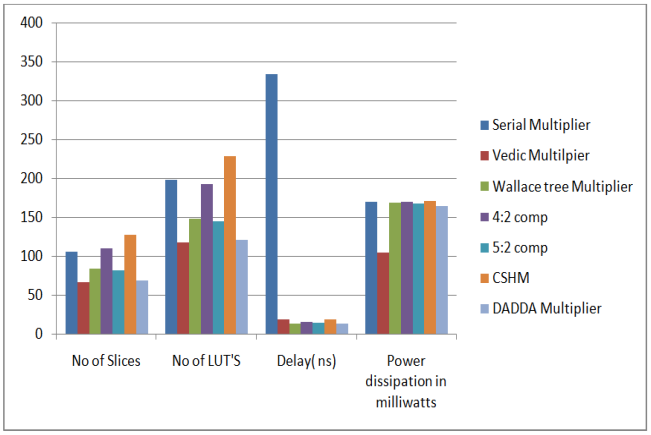

Figure 7. Comparison parameters of different multipliers.

Below table 1 summarizes the performance of all 8 bit multipliers which were implemented using Xilinx FPGA of xc3s500e-5pq208.

Table 1

Parameter	Serial Multiplier	Vedic Multiplier	Wallace tree Multiplier	4:2 comp	5:2 comp	CSHM	DADDA Multiplier
No of Slices	106	66	84	110	82	128	69
No of LUT'S	198	118	148	193	145	229	121
Delay(ns)	334.385	18.922	13.4	15.427	14.213	18.243	13.237
Power dissipation in milli watts	170.5	105	169	170	168	171	165

Figure 8 depicts the comparison of power delay product of different multipliers for each and every multipliers.

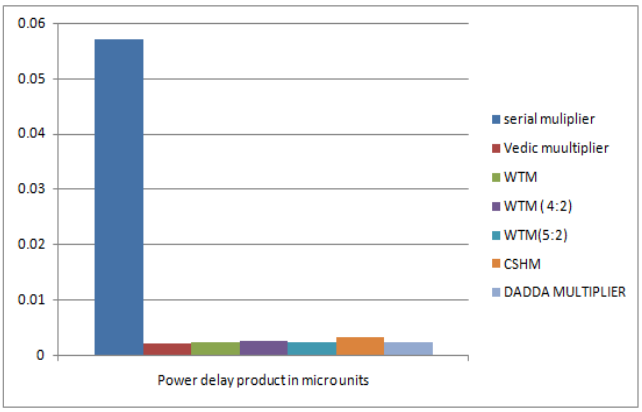

Figure 8. Comparison of power delay product of different multipliers.

It is observed that the vedic multiplier is having less power delay product. This comparison is for 8 bit unsigned binary numbers only however, it can be implemented for higher bits such 16/32 bits etc.

IV. CONCLUSIONS

Various multipliers have been simulated and synthesized using Spartan3E xc3s500e-4pq208 FPGA board. The observation for their area, delay and power consumption is studied. We have observed that the fastest multiplier for 8 bit unsigned multiplications is Vedic multiplier with less area and less power consumption. Computational Sharing multiplier is better for more number of bits multiplications like 128 bit or 256 bit because all calculations are executed in parallel, so it is efficient for large data manipulations. For small bits upto 16 bit calculations, Vedic multiplier is the best suitable multiplier as it consumes less power and requires less area. For serial calculations, like in data communications Serial Add and shift multiplier is the better choice because it requires less size and can be synchronized with the clock. If all the multipliers are needed to implement with only full adders and half adders then Wallace tree is a suitable multiplier. If we want to reduce the number of Full adders and Half adders we can use the Dadda multiplier which is simply the modification of selecting inputs to the full adders. If we need more reduction of area in Wallace tree multiplier we can replace full adders i.e. 3:2 compressors with 4:2 or 5:2 compressors. Depending on the area constraints, power constraints, bit width and clock unit we can select any one of the multipliers. Further, presented 8 bit width implementation can be extended to 16/32/64 bits for better comparisons of their performances.

REFERENCES

[1] L. Dadda, "Some schemes for parallel multipliers," Alta Frequenza, vol.34, pp. 349–356,1965.1982.

[2] P. V. Rao, Cyril Prassana Raj P,S. Ravi, "VLSI Design and Analysis of Multipliers for Low Power", IEEE 2009 Fifth International Conference On Intelligence Information Hiding and Multimedia Signal processing, ISBN: 978-1-4244-4717-6pp. 1354-1357, Sept.2009

[3] Soojin Kim, Kyeongsoon Cho," Design of High-speed Modified Booth Multipliers Operating at GHz Ranges", World Academy of Science ,Engineering and Technology, Vol:4, Jan 2010.

[4] N. Ravi, A.Satish , Dr. T. Jayachandra Prasad, Dr. T. Subba Rao,"A New Design for Array multiplier with Trade Off in power and Area", IJCSI, ISSN (Online): 1694-0814,vol. 8,issue 3, No. 2, Page No. 533to537,May 2011.

[5] Dursun Baran, Mustafa Aktan and Vojin G. Oklobdzija," Multiplier Structures for Low Power Applications in Deep-CMOS", IEEE International Symposium on Circuits and Systems (ISCAS), ISSN :0271-4302 Page(s): 1061 – 1064,May 2011.

[6] S. A. Shinde, R. K. Kamat," FPGA based Improved Hardware Implementation of Booth Wallace Multiplier using Handel C", LEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392 – 1215, 2011. No. 3(109) ,Page No.71-74, 2011.

[7] P. Verma, K.K. Mehta, "Implementation of an efficient multiplier based on Vedic Mathematics using EDA Tool", International journal of Engineering and Advance Technology (IJEAT) ISSN : 2249-8958. Volume-1, Issue-5, June 2012.

[8] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filter with powers-of-two coefficients," IEEE Trans. Circuits Syst., vol. 36, pp. 1044–1047, July 1989.

[9] K. Sethi, R. Panda, "An Improved squaring circuit for binary Numbers", International Journal of Advanced computer science and Application (IJACSA), Vol. 3, No.2, Page No.111-116, 2012.

[10] M Poornima, S. K. Patil, S. Kumar, K.P. Shridhar, H. Sanjay "Implementation of multiplier using Vedic Algorithm", International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Vol.2, Issue-6, May 2013.

[11] Asmita Haveliya :" A Novel Design for High Speed Multiplier for Digital Signal Processing Applications "(Ancient Indian Vedic mathematics approach),International Journal of Technology And Engineering System(IJTES):Vol.2,No.1,Page No.27-31,Jan – March 2011.

[12] Kantamaneni.Sravanthi, Dr.V.V.K.D.V.Prasad Battula, Veera Vasantha Rao, "Design of FIR Filter by using Sharing Multiplier with Low Delay" International Refereed Journal of Engineering and Science (IRJES),ISSN (Print) 2319-1821 Volume 1, Issue 1, PP.031-038,September 2012.

[13] Purushottam D. Chidgupkar and Mangesh T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing", 7th UICEE Annual Conference on Engineering Education,Global J. of Engng. Educ., Vol.8, 2012.

[14] P. D. Chidgup kar and M. T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing", Global J. of Engg. Edu, vol.8, no.2, 2004.

[15] Premananda B.S., Samarth S. Pai, Shashank B.,Shashank S. Bhat, "Design and Implementation of 8-Bit Vedic Multiplier", International Journal of Advanced Research in Electrical, Electronic and Instrumentation Engineering,ISSN (Print) : 2320 – 3765,Vol.2,Issue 12, December 2013.

[16] R. Shridevi, Anirudh Palakurthi, Akhila Sadhula, Hafsa Mahreen, "Design of a High Speed Multiplier (Ancient Vedic Mathematics Approach) International Journal of Engineering Research, ISSN : 2319-6890,Vol.2, Issue No.3,pp:183-18,July 2013.