Novel High Speed Vedic Mathematics Multiplier using Compressors

Sushma R. Huddar and Sudhir Rao Rupanagudi
WorldServe Education,
Bangalore, India
{sushma,sudhir}@worldserve.in

Kalpana M.
Dept. of ECE,
RYMEC
Bellary, India
kalpanamaddineni87@gmail.com

Surabhi Mohan
Dept. of Telecom,
Atria Institute of Technology,
Bangalore, India
surabhimohan06@gmail.com

***Abstract*—With the advent of new technology in the fields of VLSI and communication, there is also an ever growing demand for high speed processing and low area design. It is also a well known fact that the multiplier unit forms an integral part of processor design. Due to this regard, high speed multiplier architectures become the need of the day. In this paper, we introduce a novel architecture to perform high speed multiplication using ancient Vedic maths techniques. A new high speed approach utilizing 4:2 compressors and novel 7:2 compressors for addition has also been incorporated in the same and has been explored. Upon comparison, the compressor based multiplier introduced in this paper, is almost two times faster than the popular methods of multiplication. With regards to area, a 1% reduction is seen. The design and experiments were carried out on a Xilinx Spartan 3e series of FPGA and the timing and area of the design, on the same have been calculated.**

Keywords—4:2 Compressor, 7:2 Compressor, Booth's multiplier, high speed multiplier, modified Booth's multiplier, Urdhwa Tiryakbhyam Sutra, Vedic Mathematics

I. INTRODUCTION

The speed of a processor greatly depends on its multiplier's performance. This in turn increases the demand for high speed multipliers, at the same time keeping in mind low area and moderate power consumption [1]. Over the past few decades, several new architectures of multipliers have been designed and explored. Multipliers based on the Booth's [2] and modified Booth's algorithm [3] is quite popular in modern VLSI design but come along with their own set of disadvantages. In these algorithms, the multiplication process, involves several intermediate operations before arriving at the final answer. The intermediate stages include several comparisons, additions and subtractions which reduce the speed exponentially with the total number of bits present in the multiplier and the multiplicand [4]. Since speed is our major concern, utilizing such type of architectures is not a feasible approach since it involves several time consuming operations.

In order to address the disadvantages with respect to speed of the above mentioned methods, [5] and [6] explored a new approach to multiplier design based on ancient Vedic Mathematics. Vedic Mathematics is an ancient and eminent approach which acts as a foundation to solve several mathematical challenges encountered in the current day scenario. Vedic Mathematics existed in ancient India and was rediscovered by a popular mathematician, Sri Bharati Krishna Tirthaji [7]. He bifurcated Vedic mathematics into 16 simple sutras (formulae). These Sutras deal with Arithmetic, Algebra, Geometry, Trigonometry, Analytical Geometry etc. The simplicity in the Vedic mathematics sutras paves way for its application in several prominent domains of engineering like Signal Processing, Control Engineering and VLSI [8].

One of the highlights of the Vedic maths approach is that the calculation of all the partial products required for multiplication, are obtained well in advance, much before the actual operations of multiplication begin. These partial products are then added based on the Vedic maths algorithm to obtain the final product. This in turn leads to a very high speed approach to perform multiplication [9]. In this paper, we explore a novel method to further enhance the speed of a Vedic mathematics multiplier by replacing the existing full adders and half adders of the Vedic mathematics based multipliers with compressors. Compressors, in its several variants, are logic circuits which are capable of adding more than 3 bits at a time as opposed to a full adder and capable of performing this with a lesser gate count and higher speed in comparison with an equivalent full adder circuit [10].

Section II deals with the Urdhwa Tiryakbhyam method of multiplication using Vedic maths in detail. Section III describes the compressor architecture variants and introduces a novel compressor architecture. Section IV deals with the novel approach of combining the Vedic maths methodologies for multiplication and compressor techniques for a high speed multiplication design. Section V and VI deal with the results and future work possible with these techniques.

II. VEDIC MATHS - URDHWA TIRYAKBHYAM SUTRA

As mentioned earlier, Vedic Mathematics can be divided into 16 different sutras to perform mathematical calculations. Among these the Urdhwa Tiryakbhyam Sutra is one of the most highly preferred algorithms for performing multiplication. The algorithm is competent enough to be employed for the multiplication of integers as well as binary numbers. The term "Urdhwa Tiryakbhyam" originated from 2 Sanskrit words Urdhwa and Tiryakbhyam which mean

978-1-4673-5090-7/13/$31.00 ©2013 IEEE

"vertically" and "crosswise" respectively [7]. The main advantage of utilizing this algorithm in comparison with the existing multiplication techniques, is the fact that it utilizes only logical "AND" operations, half adders and full adders to complete the multiplication operation. Also, the partial products required for multiplication are generated in parallel and apriori to the actual addition thus saving a lot of processing time.

Let us consider two 8 bit numbers X_7-X_0 and Y_7-Y_0, where 0 to 7 represent bits from the Least Significant Bit (LSB) to the Most Significant Bit (MSB). P_0 to P_{15} represent each bit of the final computed product. It can be seen from equation (1) to (15), that P_0 to P_{15} are calculated by adding partial products, which are calculated previously using the logical AND operation. The individual bits obtained from equations (1) to (15), in turn when concatenated produce the final product of multiplication which is depicted in (16).The carry bits generated during the calculation of the individual bits of the final product are represented from C_1 to C_{30}. The carry bits generated in (14) and (15) are ignored since they are superfluous.

$$P_0 = A_0 * B_0 \tag{1}$$

$$C_1P_1 = (A_1 * B_0) + (A_0 * B_1) \tag{2}$$

$$C_3C_2P_2 = (A_2 * B_0) + (A_0 * B_2) + (A_1 * B_1) + C_1 \tag{3}$$

$$C_5C_4P_3 = (A_3 * B_0) + (A_2 * B_1) + (A_1 * B_2) + (A_0 * B_3) + C_2 \tag{4}$$

$$C_7C_6P_4 = (A_4 * B_0) + (A_3 * B_1) + (A_2 * B_2) + (A_1 * B_3) + (A_0 * B_4) + C_3 + C_4 \tag{5}$$

$$C_{10}C_9C_8P_5 = (A_5 * B_0) + (A_4 * B_1) + (A_3 * B_2) + (A_2 * B_3) + (A_1 * B_4) + (A_0 * B_5) + C_5 + C_6 \tag{6}$$

$$C_{13}C_{12}C_{11}P_6 = (A_6 * B_0) + (A_5 * B_1) + (A_4 * B_2) + (A_3 * B_3) + (A_2 * B_4) + (A_1 * B_5) + (A_0 * B_6) + C_7 + C_8 \tag{7}$$

$$C_{16}C_{15}C_{14}P_7 = (A_7 * B_0) + (A_6 * B_1) + (A_5 * B_2) + (A_4 * B_3) + (A_2 * B_5) + (A_1 * B_6) + (A_0 * B_7) + C_9 + C_{11} \tag{8}$$

$$C_{19}C_{18}C_{17}P_8 = (A_7 * B_1) + (A_6 * B_2) + (A_5 * B_3) + (A_4 * B_4) + (A_3 * B_5) + (A_2 * B_6) + (A_1 * B_7) + C_{10} + C_{12} + C_{14} \tag{9}$$

$$C_{22}C_{21}C_{20}P_9 = (A_7 * B_2) + (A_6 * B_3) + (A_5 * B_4) + (A_4 * B_5) + (A_3 * B_6) + (A_2 * B_7) + C_{13} + C_{15} + C_{17} \tag{10}$$

$$C_{25}C_{24}C_{23}P_{10} = (A_7 * B_3) + (A_6 * B_4) + (A_5 * B_5) + (A_4 * B_6) + (A_3 * B_7) + C_{16} + C_{18} + C_{20} \tag{11}$$

$$C_{27}C_{26}P_{11} = (A_7 * B_4) + (A_6 * B_5) + (A_5 * B_6) + (A_4 * B_7) + C_{19} + C_{21} + C_{23} \tag{12}$$

$$C_{29}C_{28}P_{12} = (A_7 * B_5) + (A_5 * B_6) + (A_5 * B_7) + C_{22} + C_{24} + C_{26} \tag{13}$$

$$C_{30}P_{13} = (A_7 * B_6) + (A_6 * B_7) + C_{25} + C_{27} + C_{28} \tag{14}$$

$$P_{14} = (A_7 * B_7) + C_{29} + C_{30} \tag{15}$$

$$P_{15} = (A_7 * B_7) \tag{16}$$

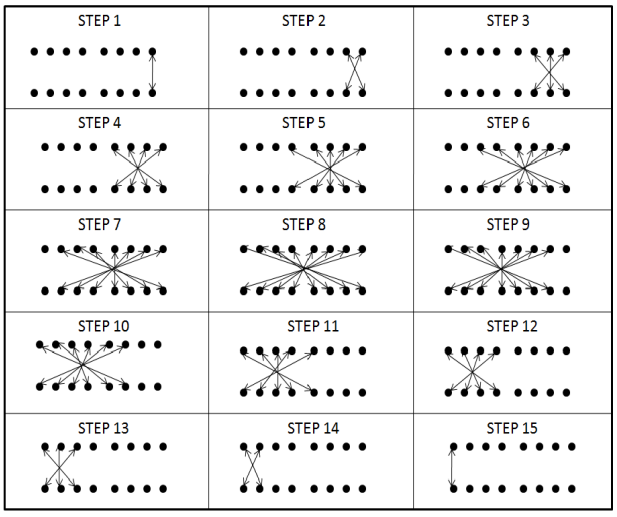


Fig. 1. Pictorial Illustration of Urdhwa Tiryakbhyam Sutra for multiplication of 2 eight bit numbers

Fig.1 graphically illustrates the step by step method of multiplying two 8 bit numbers using the Urdhwa Tiryakbyam Sutra. The black circles indicate the bits of the multiplier and multiplicand, and the two-way arrows indicate the bits to be multiplied in order to arrive at the individual bits of the final product. The hardware architecture of the 8x8 Urdhwa multiplier has been designed and shown in Fig. 2.

As mentioned earlier, the partial products obtained are added with the help of full adders and half adders. It can be seen, from equation (1) to (16), that in few equations there is a necessity of adding more than 3 bits at a time. This leads to additional hardware and additional stages, since the full adder is capable of adding only 3 bits at a time. In the next section two different types of compressor architectures are explored which assist in adding more that 3 bits at a time, with reduced architecture and increased efficiency in terms of speed.

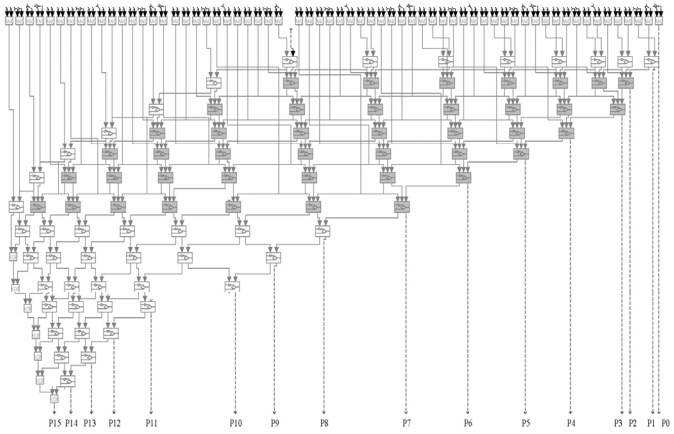


Fig. 2. Hardware architecture of Urdhva Tiryakbhyam multiplier where the gray colored blocks represent the full adders and the white colored blocks represent the half adders

.

III. COMPRESSOR ADDER

A compressor adder is a logical circuit which is used to improve the computational speed of the addition of 4 or more bits at a time. Compressors can efficiently replace the combination of several half adders and full adders, thereby enabling high speed performance of the processor which incorporates the same. The compressor adder used in this paper is a 4:2 compressor adder. A lot of research in the past has been carried out on the same [11]. This has been elaborated below. A comparison of the 4:2 compressor with an equivalent circuit, using full adders and half adders has also been given below.

A. 4:2 Compressor Adder

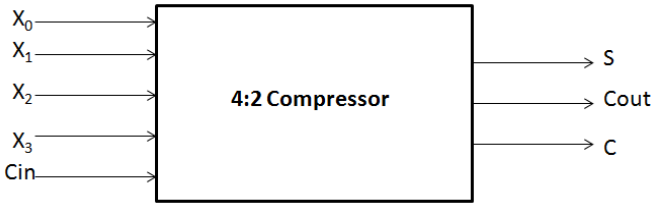


Fig. 3. Black box of a 4:2 compressor adder

A 4:2 compressor as shown in fig.3. is capable of adding 4 bits and one carry, in turn producing a 3 bit output. The internal architecture of the same has been show in fig.4. It can be clearly seen, the critical path is smaller in comparison with an equivalent circuit to add 5 bits using full adders and half adders. For the sake of comparison, the equivalent circuit to add 5 bits has also been shown in Fig. 5.

Let us consider the propagation delay of a gate to be tp. It is well known that a full adder has a total propagation delay of 2tp and a half adder has a propagation delay of tp. Considering this, the total propagation delay of a 4:2 adder using full adders and half adders can be calculated as 5tp and can be seen in Fig.5. On the other hand, it can be seen from Fig. 4. that the propagation delay of a 4:2 compressor remains only 3tp. Therefore, a 66.6% increase in speed can be recorded in comparison with an equivalent circuit made of full and half adders, proving to be a highly efficient architecture for addition.

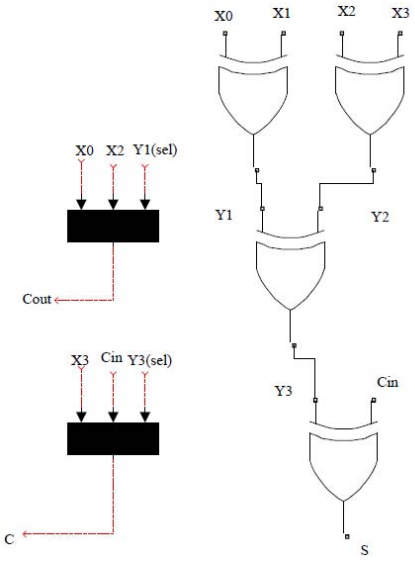


Fig. 4. Gate level diagram of 4:2 Compressor

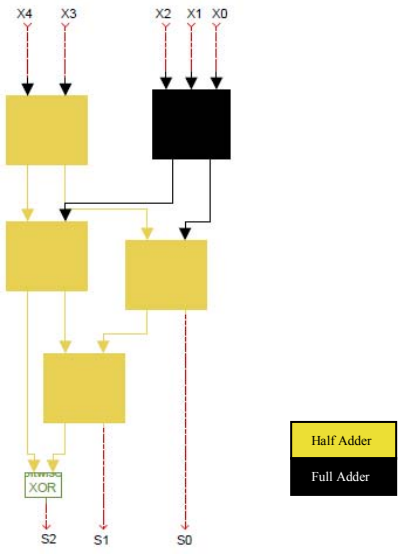


Fig. 5. 4:2 Compressor using full adders and half adders

In order to add more than 5 bits at a time, yet another compressor architecture – the 7:2 compressor adder could be used and this is explained in detail below.

B. 7:2 Compressor Adder

Similar to its 4:2 compressor counterpart, the 7:2 compressor as shown in Fig. 6., is capable of adding 7 bits of input and 2 carry's from the previous stages, at a time. In our implementation, we have designed a novel 7:2 compressor utilizing two 4:2 compressors, two full adders and one half adder. The architecture for the same has been shown in Fig. 7. As mentioned earlier, since the 4:2 compressor shows a significant increase in speed by around 66.6%, utilizing the same in this architecture would improve the efficiency as opposed to a conventional approach of adding nine bits at a time using only full adders and half adders. This leads to a great improvisation in speed of the processor. Through experimentation on a Xilinx Spartan-3e FPGA, it was found that the novel 7:2 compressor adder architecture introduced here is 1.05 times faster than a conventional approach. This result justifies the need of utilizing this compressor in our design.

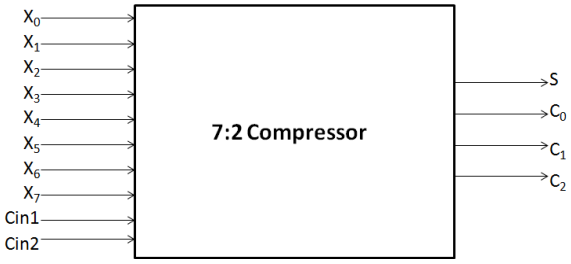


Fig. 6 Black box representation of a 7:2 Compressor Adder

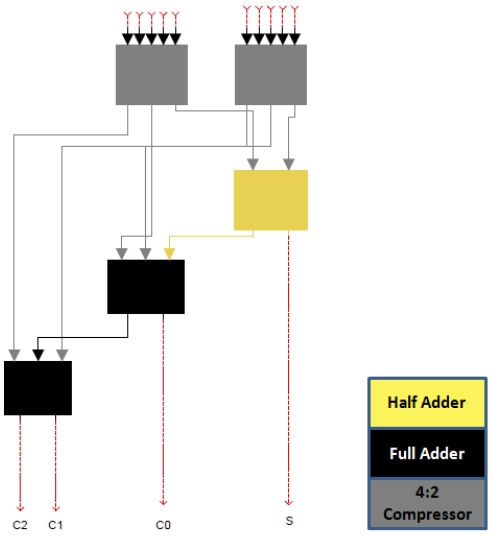


Fig.7 7:2 Compressor using 4:2 Compressor adder

The next section discusses a novel approach to combine the efficiency of the compressor architectures introduced above with the existing Vedic mathematics approach for multiplication.

IV. Compressor Based Urdhwa Tiryakbhyam Multiplier

As mentioned in Section II, the multiplier based on Urdhwa method of multiplication requires several full adders and half adders to add the necessary partial products. This in turn leads to a large propagation delay due to the reasons explained in the previous section. As part of our novel approach, we combined the compressor architectures explained earlier and utilized the same in the Urdhwa based architecture which was formerly shown in Fig.2. The architecture for the same has been shown below in Fig.8.

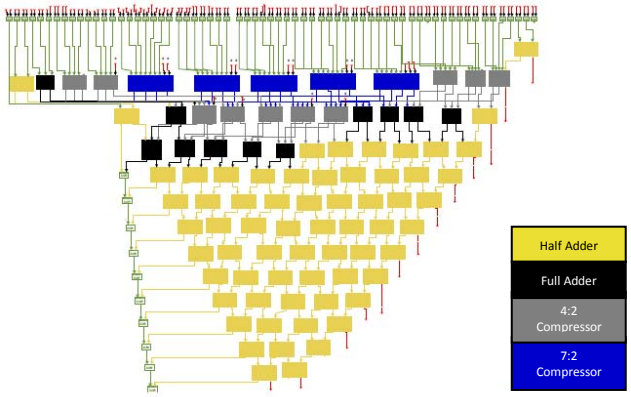


Fig.8 Hardware architecture of Compressor based Urdhwa Multiplier

It can be clearly seen from Fig. 8 that the compressor based Urdhwa multiplier requires only 12 parallel stages as opposed to 15 which was in the case of the conventional Urdhwa Tiryakbhyam multiplier. This is a major improvement with respect to high speed multiplier design. Also, it can be seen that, many of the stages have now been reduced to a mere logical XOR operation, with an initiative to reduce area.

An analysis on the area occupied by the new design and also the improvement in speed in comparison with other popular methods of multiplication has been presented in the next section.

V. Results

In order to perform a comparison, various popular multipliers – Booth's, modified Booth's, Urdhwa multiplier and also the compressor based Urdhwa multiplier were implemented on a Xilinx Spartan 3e – XC3S100E FPGA using Verilog as the RTL language with the help of Xilinx Project Navigator 12.1. The codes were synthesized. Unoptimized speed and area parameters were compared. The Spartan 3e FPGA used for the experiments has a speed grade

of -5 and package CP132. The results have been tabulated in Table 1.

TABLE I. COMPARISON OF AREA OCCUPIED AND SPEED OF VARIOUS MULTIPLIER ARCHITECTURES

Algorithm used	LUTs Used	Total LUTs present	% of area occupied	Frequency (MHz)	Time (ns)
Booth algorithm	55	1920	2	32.84	30.448
Modified Booth algorithm	213	1920	11	45.982	21.747
Urdhwa-Tiryakbhyam	185	1920	9	61.869	16.163
Compressor based Urdhwa Tiryakbhyam	170	1920	8	69.153	14.41

It can be clearly noted from Table I., that in terms of speed, the compressor based Vedic maths multiplier performs exceptionally well and is almost 1.12 times faster than the existing Vedic maths based multiplier. It can also be seen that in comparison with the booth and modified booth multipliers, the new architecture is around 2.112 times and 1.509 times faster respectively with regards to speed.

Another interesting thing to note is the area occupied. Since, the 4:2 compressor has reduced number of gates as compared to a full/half adder based circuit, the area has also reduced equivalently. It can be seen that the compressor based multiplier has occupied an area 1% lesser than the Vedic maths and a 3% reduction with respect to the modified booth methodology is also seen. Even though the compressor based architecture occupies area more than that of the booths method, since speed is our major concern, this fact can be ignored.

VI. CONCLUSION

In this paper, we have proposed a novel high speed architecture for multiplication of two 8 bit numbers, combining the advantages of compressor based adders and also the ancient Vedic maths methodology. A new 7:2 compressor architecture, based on 4:2 compressor architecture was also discussed. Upon comparison of the area occupied by the multiplier and also its speed, with two other popular multipliers, we can conclude that the compressor based Vedic maths multiplier proves to be a better option over conventional multipliers used in several expeditious and complex VLSI circuits. As a future work, the multiplier's performance could be tested within an ALU and also compared with several other existing multipliers.

REFERENCES

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

[2] A.D.Booth, "A Signed Binary Multiplication Technique," J. mech. and appl. math, vol 4, no.2, pp. 236-240, Oxford University Press, 1951.

[3] C. R. Baugh, B. A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm,", IEEE Trans. Computers 22(12), pp. 1045–1047, 1973.

[4] Koren Israel, "Computer Arithmetic Algorithms," 2nd Ed, pp. 141-149, Universities Press, 2001.

[5] L. Sriraman, T.N. Prabakar, "Design and Implementation of Two Variable Multiplier Using KCM and Vedic Mathematics," 1st Int. Conf. on Recent Advances in Information Technology, Dhanbad, India, 2012, IEEE Proc., pp. 782-787.

[6] M. Ramalatha, K. Deena Dayalan, S. Deborah Priya, "High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques," Advances in Computational Tools for Engineering Applications, 2009, IEEE Proc., pp 600-603.

[7] Jagadguru Swami Sri Bharati Krisna Tirthaji Maharaja, "Vedic Mathematics: Sixteen Simple Mathematical Formulae from the Veda," pp. 5-45, Motilal Banarasidas Publishers, Delhi, 2009.

[8] Himanshu Thapliyal and M. B. Srinivas, "An efficient method of elliptic curve encryption using Ancient Indian Vedic Mathematics," 48th IEEE Int. Midwest Symp. on Circuits and Systems, 2005, vol. 1, pp. 826-828.

[9] Tiwari, Honey Durga, et al., "Multiplier design based on ancient Indian Vedic Mathematics, " Int. SoC Design Conf., 2008, vol. 2. IEEE Proc., pp. II-65 - II-68.

[10] Hsiao, Shen-Fu, Ming-Roun Jiang, and Jia-Sien Yeh, "Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers," IEEE Electronics Letters, vol. 34, no.4, pp. 341-343, Feb. 1998.

[11] D. Radhakrishnan and A. P. Preethy, "Low power CMOS pass logic 4-2 compressor for high-speed multiplication," Circuits and Systems, Proc. 43rd IEEE Midwest Symp., vol. 3, pp. 1296-1298, 2000.