2nd IEEE International Conference on Engineering and Technology (ICETECH), 17th & 18th March 2016, Coimbatore, TN, India.

VLSI Design of High Speed Vedic Multiplier for FPGA Implementation

Kapil Ram Gavali[1], Poonam Kadam
Department of Electronics & Telecommunication Engineering
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
gavalikapil1989@gmail.com[1]

***Abstract*— In the modern world of digitization, processing of data in real time requires an increase in the operating speed of a system. The processing more often than not utilizes multiplication which is time consuming and introduces considerable amount of delay. As such, there is a need to reduce this delay and achieve faster real time processing of data. This paper proposes a novel architecture for implementation of signed multiplication using the vedic algorithm. An 11x8 bit multiplier was designed using the proposed architecture and implemented using Xilinx ISE Design Suite 13.2 with Spartan 3E as the target FPGA. The maximum clock speed achieved was 203.938 MHz.**

Keywords*— Vedic Multiplication, pipelining, real-time processing

I. INTRODUCTION

In the current era of "Internet of Things", efforts are being made to interconnect various objects which are embedded with electronics and sensors to facilitate exchange of information, it is very important that the underlying hardware be capable of processing real time data quickly. As such a lot of efforts are underway to design systems capable of real time operations [1] [2] [3] [4]. Many of these systems are implemented in Field Programmable Gate Arrays (FPGAs) owing to their massive parallelism. As such many traditional algorithms are being mapped in efficient architectures in such a way so as to take the advantage of inherent parallelism in an FPGA [4] [5] [6]. However the inherent area-speed-power tradeoff often exist when an architecture is designed for FPGA implementation. Often an effort to increase the processing speed would result in higher power consumption or more area. As such efficient mapping of an algorithm into architecture suitable for FPGA implementation pose a challenge.

Many real time processing applications involve the use of algorithms such as Discrete Cosine Transform which include multiplication of continuously varying incoming data. Multiplication of two numbers is a time consuming process as it involves carry propagation along with successive additions. However, it is necessary for real time applications that there be no delay in subsequent output samples. As such, there is a necessity for an efficient multiplication algorithm and architecture which provides a high throughput and can handle fast varying input data.

A lot of different algorithm exist like Wallace, Toom Cook, Booth and vedic for multiplication of two numbers. Their characteristics and performances in an FPGA based implementation have been reviewed in literature [7] [8]. Vedic multiplication was found to perform better as compared to the other algorithms. A lot of different architectures have been reported in the literature concerning the implementation of vedic algorithm [8] [9] [10].

The objective of this paper is to present a novel architecture for implementation of vedic multiplication algorithm suitable for FPGA implementation. The proposed architecture avoids the shift and add proposed in [9] and thus achieves higher speed and throughput. The remainder of the paper is organized as follows.

Section II describes the algorithm for computation of vedic multiplication. Section III describes the architecture for the algorithm. Section IV discusses the results obtained in simulation and Section V concludes the paper.

II. ALGORITHM

The algorithm to multiply two four bit numbers using Vedic Multiplication is illustrated in Fig. 1(a)-(g). The partial products S1 to S8 are calculated as indicated in the figure where a double headed arrow between two bits means bit wise logical AND. The presence of two or more arrows in a figure represents the addition of the result of logical AND between the individual arrows. The Least Significant Bit (LSB) of each of these partial products (S1[0] to S7[0]) give us the partial result. The remaining bits of these partial products are carried over to the next partial product for addition designated as C1 to C6. The carry C6 gives the MSB S8[0] of the final sum. Thus, the required result is S1[0] to S8[0]. This method can be generalised for an 11x8 multiplier where the partial products would be 18 from S1 to S18.

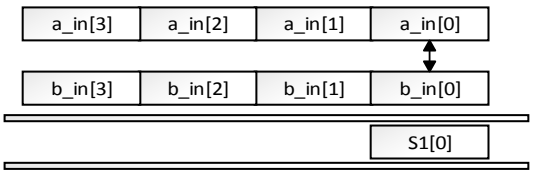

Fig. 1(a) Calculation of Partial Product S1

978-1-4673-9916-6/16/$31.00 ©2016 IEEE

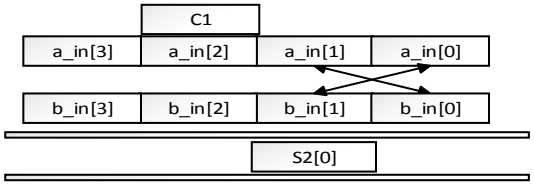

Fig. 1(b). Calculation of Partial Product S2

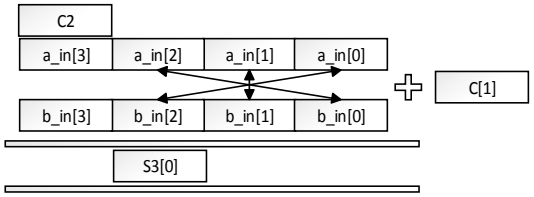

Fig. 2(c) Calculation of Partial Product S3

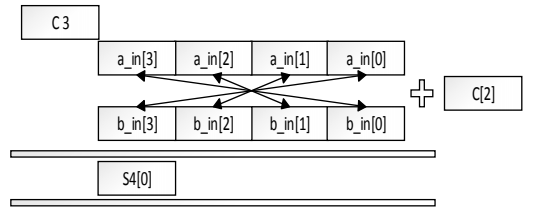

Fig. 1(d). Calculation of Partial Product S4

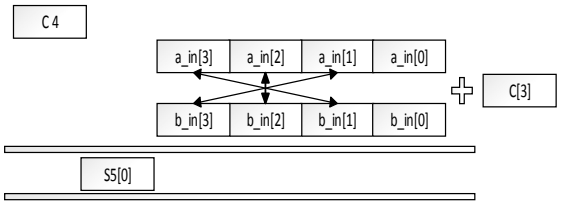

Fig. 3(e) Calculation of Partial Product S5

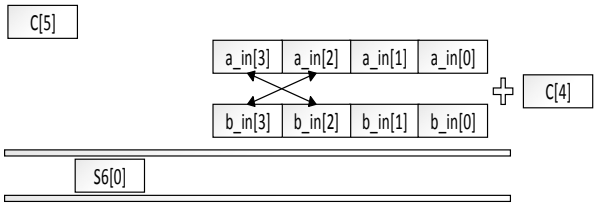

Fig. 1(f). Calculation of Partial Product S6

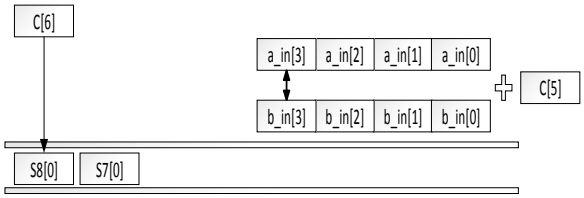

Fig. 1(g) Calculation of Partial Products S7 and S8

III. ARCHITECTURE

A highly pipelined architecture is proposed for the Vedic Multiplier implementing 4 stages which are elaborated here.

Stage I: Since the multiplication is to be performed on an FPGA which supports parallelism, the numbers to be multiplied a_in[10:0] and b_in[7:0] are available at the input simultaneously. A Logical AND operation is performed on each one of the bits of these numbers. These outputs are registered when the first clock pulse arrives so as to use them in the next stage. Since these values are now stored, this stage is free to perform AND operation on two new sets of numbers.

Stage II: The second stage computes the pre-partial products P1 to P18 as indicated in Fig. 1 by double headed arrows using logical AND operation as discussed in Section II. These are computed in a single clock pulse taking advantage of inherent parallelism provided by an FPGA. The outputs are registered at the arrival of next clock pulse for their use in the next stage and also be primed for computing the next pre-partial products coming from stage I after logical AND operation on each one of the bits.

Stage III: In the third stage, the actual required partial products S1 to S18 are computed taking the carries generated into consideration as shown in Fig. 2. 16 clock pulses are required to completely calculate the partial products S1 to S18 owing to the propagation of carry (generated from S2 to S18). There is no carry generated for S1 as only two bits a_in[0] and b_in[0] are multiplied which was shown by a single double-headed arrow in Fig. 1(a). The generated partial products are split up into their LSB and the remaining bits are used for addition with the next partial products to be computed. These remaining bits are designated as C1 to C17. Once computed, these outputs are again registered at the next clock pulse for their use in the next stage.

Stage IV: Finally, the LSBs computed in stage III are concatenated with the last carry generated to get the required result at the 19th clock pulse in the fourth stage.

The four stages described above are implemented in a nineteen stage pipeline to increase the operating speed. An architecture implemented in this way increases the operating speed as output is obtained at each and every clock pulse once the first one is obtained. This implies that in a nineteen stage pipeline, once an output is obtained, at the same time, 17 multiplications are being performed to give output at every clock pulse; thus increasing the throughput. Conventional multiplication would provide the output only after one complete multiplication.

Fig. 2. Architecture of 11*8 Vedic Multiplication

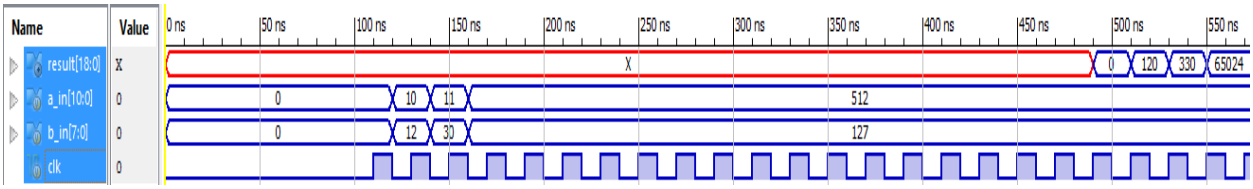

Fig. 3. Simulation Results for 11*8 Vedic Multiplier

IV. Results and Discussions

The proposed 11*8 Vedic Multiplier was simulated on a Xilinx Spartan 3E FPGA Board and subsequently synthesised to obtain the area and speed results. The same are discussed further.

The simulation results for the proposed multiplier are depicted in Fig. 3. It can be clearly seen that the first set of output is obtained at the 20th clock pulse when the clock becomes active which implies that a 19 stage pipeline was used. After this first output, the subsequent outputs are obtained at every clock pulse. As a result, this architecture can

be used in applications which require a high speed of operation.

TABLE I. COMPARISON OF DIFFERENT VEDIC MULTIPLIERS IN TERMS OF SPEED AND AREA

	Proposed Multiplier(11 *8 bits)	***Vedic Multiplier [9]***	***Vedic Multiplier [11] (4*4 bits)***	***Vedic Multiplier [12] (4*4 bits)***
Maximum Clock Speed (MHz)	203.938	179.69	37	150.76
Number of Slices	359 out of 9312 (3%)	225 out of 9312 (2%)	-	-
Total Number of LUTs	344 out of 9312	235 (all 4 inputs) out of 9312	-	833(4, 3, and 2 inputs)

The comparison of different vedic multipliers in terms of their speed and area is tabulated in Table I. It is observed that the proposed multiplier achieves a maximum clock frequency of 203.938 MHz which is an improvement of 13.49% over the work done in [9]. Even though the proposed multiplier occupies larger area, it is useful in applications where the operating speed is of essence.

V. CONCLUSION

The proposed architecture is capable to achieve high speed and throughput in processing as it avoids the shifting and adding of the partial products. The increase in the number of pipeline stages also help in increasing the throughput. Even though there is a marginal increase in the occupied area, this would not cause a major issue in FPGAs such as Spartan 3E and Kintex which have a large number of slices. Thus, this architecture would be a good candidate to be implemented as a large part of systems like DCT, FFT where high speed multiplications along with good throughput are required.

References

[1] Chiang, C., Chen, Y., Ke, K., & Yuan, S. (2015). Real-time Pedestrian Detection Technique for Embedded Driver Assistance Systems, 206–207.

[2] Fico, V. M., A' Soaje, R. (2015). Implementing the Unscented Kalman Filter on an Embedded System : a Lesson Learnt. *Industrial Technology (ICIT), 2015 IEEE International Conference on*, (1), 2010–2014.

[3] Randel, S., Corteselli, S., Badini, D., Pilori, D., Caelles, S., Chandrasekhar, S., … Road, H. (2015). First Real-Time Coherent MIMO-DSP for Six Coupled Mode Transmission, 1–2.

[4] Saha, S., & Hawlader, A. K. (2015). Dynamically Reconfigurable Parallel Architecture Implementation of 2D Convolution for Image Processing over FPGA, (May), 21–23.

[5] Oliveira, P. A. M., Cintra, R. J., Bayer, F. M., Kulasekera, S., & Madanayake, A. (2016). Low-complexity Image and Video Coding Based on an Approximate Discrete Tchebichef Transform. *IEEE Transactions on Circuits and Systems for Video Technology*, *8215*(c), 1–1. http://doi.org/10.1109/TCSVT.2016.2515378

[6] Dali, M., Gibson, R. M., Amira, A., Guessoum, A., & Ramzan, N. (2015). An Efficient MIMO-OFDM Radix-2 Single-Path Delay Feedback FFT Implementation on FPGA.

[7] P. Kasat, D. Bilaye, H. V. Dixit, R. Balwaik, and A. Jeyakumar, "Multiplication algorithms for vlsi-a review," International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 11, pp. 1761–1765, Nov 2012.

[8] Kodali, R. K., Yenamachintala, S. S., & Boppana, L. (2014). FPGA Implementation of 160- bit Vedic Multiplier, 1–5.

[9] Dixit, H. V, Kasat, P. S., Balwaik, R., & Jeyakumar, A. (2010). A Parallel Pipelined Approach to Vedic Multiplier for FPGA Implementation, 284–287.

[10] Kumar, P., & Goud, S. (2013). FPGA Implementation of high speed 8-bit Vedic multiplier using barrel shifter, 14–17.

[11] Paramasivam, M. E., Dr. Sabeenian, R. S.: An Efficient Bit Reduction Binary Multiplication Algorithm using Vedic Methods. IEEE 2nd International Advance Computing Conference (2010).

[12] Laxman P. Thakre, Suresh Balpande, Umesh Akare, Sudhi Lande: Performance Evaluation and Synthesis of Multiplier used in FFT operation using Conventional and Vedic algorithms. In: Third International Conference on Emerging Trends in Engineering and Technology (2010).