Implementation of Efficient Multiplier for High Speed Applications Using FPGA

Mohamed Barakat[1], Waleed Saad[2, 3], and Mona Shokair[3]
[1]Communication and Computer Engineering Dep., Faculty of Engineering, Nhada University, Bani-Sweif, Egypt.
[2] Electronic and Electrical Comm. Dep., Faculty of Electronic Engineering, Menoufia University, Egypt.
[3] Electrical Engineering Dep., College of Engineering, Shaqra University, Dawadmi, Ar Riyadh, Saudi Arabia.
mhabbarakat@yahoo.com, waleed.saad@el-eng.menofia.edu.eg, mona.sabry@el-eng.menofia.edu.eg

***Abstract*—Multiplication is a predominant operation in many DSP applications. Area and delay are enormous in the multiplication operations, so the high-speed multiplier with a compromised area is required. This paper submits an efficient implementation of high-speed multiplier using a mixed between Vedic mathematics and high-speed adder like Carry Save Adder (CSA). CSA is convenient in adding three numbers and sutra of Vedic mathematics called Urdhva Tiryakbhyam is very suitable for multiplication. Starting with a 2-bit Vedic multiplier, we can be ascending to implement a 64-bit multiplier. After comparison, the obtained result from the proposed multiplier is better than the references in terms of area and delay. All algorithms are coded in VHDL and targeted to implement on Virtex-5 and Virtex-6 FPGA kit with ISE 14.5, the simulated results are acquired by Modelsim 10.3d.**

Keywords—Carry Save Adder; Vedic mathematics; Urdhva-Tiryakbhyam; Implementation.

I. INTRODUCTION

Addition, division, subtraction, and multiplication are considered as the main Arithmetic operations. As compared to other operations like Addition and subtraction, multiplication has a massive delay and area. The multiplier is the predominant module of various DSP processors. In many applications like graphics acceleration and DSP application, FFT etc. [1, 3]. The multiplier must be having a low delay and a low area, so many efforts have been made over a few decades to enhance the multiplication speed. Multiplication operations are not only complex but also needs more power consumption. The parameters like delay, power, and area must be optimized, so swapping between all these parameters must be achieved to obtain the in-demand performance.

In this paper, a mix between Urdhva-Tiryakbhyam (Vedic Mathematics) and fast adder such as Carry Cave Adder has been presented. this combination reduces the delay and decreases the hardware resources as compared to familiar methods [1, 7]. The designs are carried out on Virtex-5 FPGA kit with ISE 14.5, and the simulated results are verified by Modelsim 10.1c.

The rest of paper is orderly as follows: Section II shows the operation of Carry Save Adder, Section III explain the Urdhva Tiryakbhyam Algorithm, Section IV shows the proposed multiplier techniques, Section V shows the Experimental Results and Finally, Conclusion in Section VI.

II. CARRY SAVE ADDER

CSA applied in the microarchitecture of the computer to get the sum of more n-bit numbers. The output of CSA is two numbers, one which is a sequence of partial sum m_n bits and the other is a sequence of carry bits n_n as shown in figure 1. The CSA consists of a ladder of full adders depending on the number of bits added [1]. The summing operation of three numbers using CSA for 4-bit length and using RCA to obtain the final sum is shown below

An example of summing three numbers using CSA:

a		**1**	**0**	**1**	**1**
b	**+**	**1**	**0**	**0**	**1**
c	**+**	**0**	**1**	**1**	**1**
m		**0**	**1**	**0**	**1**
n	**1**	**0**	**1**	**1**	
SUM	**1**	**1**	**0**	**1**	**1**

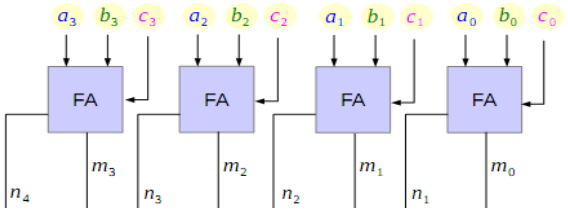

Fig. 1. Three input Carry Save Adder

III. URDHVA TIRYAGBHYAM ALGORITHM

Urdhva Tiryakbhyam is an old Vedic mathematics, 'Urdhva' means 'vertically' and 'Tiryakbhyam' means 'crosswise' Fig. 2 shows an example for using Vertically and Crosswise Multiplication [1, 3, 5, 6]. The Vedic algorithm is suitable to the multiplication operations of any numbers of bits as it submits a parallel enforcement of a partial product and then sums these partial products to get the final product.

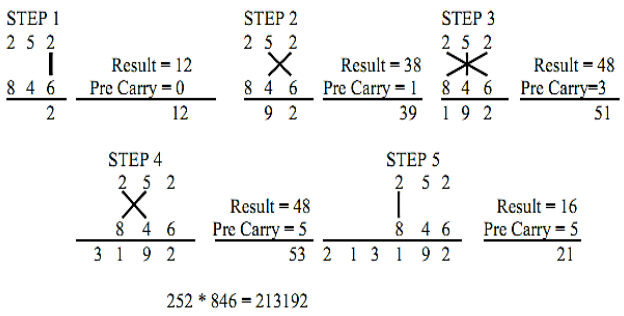

Fig.2. Vertically and Crosswise Multiplication of two decimal numbers.

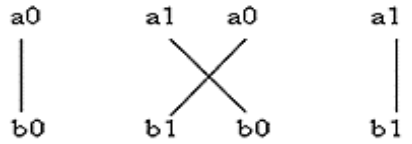

Fig. 3. 2-bit binary numbers Vedic multiplication

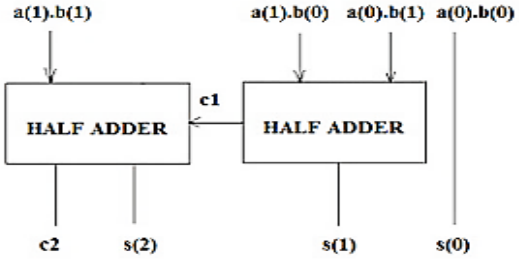

Fig. 4: Implementation method for 2-bit Vedic Multiplication.

Now we introduce the operation of the Vedic Multiplier in a binary number. Assume A and B are 2-bit binary numbers where $A = a_1a_0$ and $B = b_1b_0$ as showed in Fig. 3. The operation starts with the vertical multiplication of Least Significant Bit (LSB) of the two number a_0 and b_0 , the result S_0 is acceptable as the LSB of the final product. After that, the crosswise multiplication of LSB of A with the Most Significant Bit (MSB) of B and vice versa followed by adding the two products [4, 10, 11]. The summation of the result of crosswise multiplication gives the second bit S_1 of the final product and the carry C_1 is added to the earlier vertical product acquires by multiplying the MSB's, which leads the sum S_2 and carry C_2 terms. The sum and carry act as corresponding third and fourth final product term.

$$S_0 = a_0b_0 \quad (1)$$

$$C_1S_1 = a_1b_0 + a_0b_1 \quad (2)$$

$$C_2S_2 = C_1 + a_1b_1 \quad (3)$$

The final product term is $C_2S_2S_1S_0$. This process is applicable to any number of bits. Vedic multiplication of 2*2-bit is designed by two half-adders and four AND gates as shown below in Figure 4.

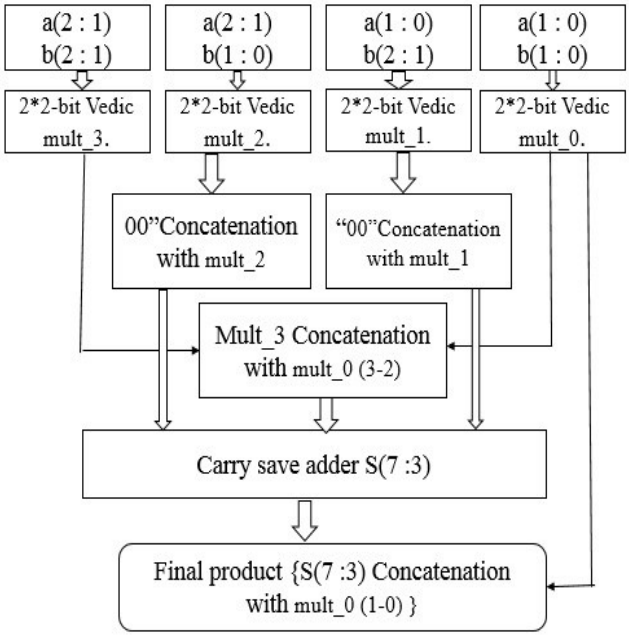

Fig. 5. Proposed 4*4-bit Vedic Multiplier.

IV. THE PROPOSED MULTILIER

The proposed N*N-bit multiplication that using a mixed between Vedic mathematics and CSA is showing in this section. The 4*4-bit proposed multiplication design is introduced firstly, after that, we can ascend to design any N*N-bit multiplication.

*A. 4*4 Bit proposed multiplier*

The proposed 4*4-bit multiplication shows in Fig.5. Assume two binary numbers A and B where $A = a_3a_2a_1a_0$ and $B = b_3b_2b_1b_0$. The two numbers A and B divide into two equal halves, $A = A_mA_l$ and $B = B_mB_l$ where m and l stands for MSB and LSB respectively.

the conventional method for Vedic multiplier. We have three adder and four 2*2-bit multiplication [1, 3, 6]. The proposed multiplier overcoming the three adders by using one CSA and one implemented RCA to get the final sum. The partial products are $\{mult\_0 = A_lB_l$, $mult\_1 = A_l * B_m$, $mult\_2 = A_m * B_l$ and $mult\_3 = A_m * B_m\}$ each multiplier has 4-bit length as shows in Fig. 5. Firstly The 4-bit of mult_3 are concatenated with the most significant 2-bits of mult_0, to get 6-bit length. Secondly for mult_1 and mult_2 we insert two zeros concatenation in left hand side to get 6-bit length. Now we have three number each one has 6-bit length which can be added directly by CSA as shown in Fig. 5. The final sum term can be made by RCA that has a 6-bit length (S7-S3) concatenated with least significant 2-bit of mult_0 to get the final product which is 8-bit length.

From Fig. 5. the Proposed 4*4 Bit Multiplier is implemented by four 2*2-bit Vedic multiplier and one CSA and one RCA Which the speed of the multiplier comes from, and the area will be reduced as well.

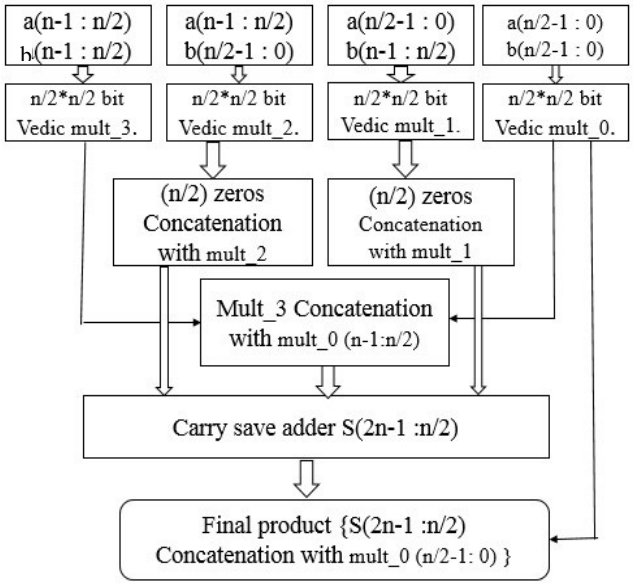

Fig. 6. Proposed N*N-bit Multiplier with CSA.

*B. The Proposed N*N-Bit Multiplier*

We develop the multiplier of 4*4 bit from the available 2*2-bit multiplier, furthermore, the 4*4-bit multiplier is optimizing. By the same way, the 8*8-bit multiplier develops fromathe multiplieraof 4*4-bit and the 16*16-bit multiplier develops from the 8*8-bit multiplier and so on [1]. Fig. 6 shows the architecture of the proposed N*N-bit multiplier. Using CSA to add three numbers is faster than using conventional three Ripple Carry Adder [6].

V. EXPERIMENTAL RESULTS

In this work, the hardware platforms of a mix between Vedic mathematics and CSA have been carried out. The RTL codes are written by very high-speed integrated circuit hardware description language (VHDL). Xilinx ISE 14.5 is used to implement the designs on Virtex-5 Field Programmable Gate Array (FPGA). The simulations are done using Modelsim 10.3d and verified the results by Matlab software.

The area and speed are our metric parameters. Table I shows the summary of the implementation results for 4, 8, 16, 32 and 64-bit. To prove the efficiency of the proposed multiplier based on the 64-bit multiplier, the Mapping report has been compared with the previously designed 64-bit multiplier. Table II, Table III and Table IV shows the speed of proposed Vedic multiplier for any number of bits when compared to the references. Table V shows that the proposed multiplier reduces the hardware resources (no. of LUTs) compared to the fast multiplier [8]. The RTL schematic for 4-bit is appeared in Fig.7.and the design summary of Map report for 64-bit is shown in Fig.8. The simulation result for 16 and 32-bit are shown in Fig.9-a.and Fig.9-b respectively.

TABE I. SUMMARY OF PROPOSED VEDIC MULTIPLIER WITH CSA ON VIRTEX-6 ML605

No. of Bits	**4-bit**	**8-bit**	**16-bit**	**32-bit**	**64-bit**
No. of slices (LUTs)	20	96	423	1771	7243
No. of occupied Slices	9	41	165	678	2879
Delay (ns)	2.926	4.651	6.612	8.779	11.432
No. of bonded IOBs	16	32	64	128	256
Average Fanout	4.2	3.91	3.89	3.92	3.95
Levels of Logic	7	18	37	72	139

TABLE II. DELAY (NS) COMPARISON AT 64-BIT

Module	**Virtex-5 ML510**	**Virtex-6 ML605**
Ref. [1]	21.985	15.770 ns
Ref. [7]	21.243	12.717
Proposed multiplier	19.615	11.432

TABLE III. DELAY (NS) COMPARISON

Module Spartan 3E	**8-bit**	**16-bit**
Ref. [8] Montgomery Multiplier	38.801	78.232
Proposed multiplier	13.532	17.852

TABLE IV. DELAY (NS) COMPARISON

Module Virtex 4 XC4vfx140	**32-bit**	**64-bit**
Ref. [9] MBA with WT	27.95	46.11
Proposed multiplier	18.718	23.792

TABLE V. AREA COMPARISON (LUTs) AT 64-BIT

module	**Virtex-6 ML605**
Ref. [7]	8185
Proposed multiplier	7243

VI. CONCLUSION

The hardware platforms of a mix between Vedic mathematics and CSA have been implemented in this paper. Designing 2^n*2^n multiplier from $2^{n-1}*2^{n-1}$ This method allows us to design a multiplier of any 2^n-bit number. From the comparative results, it's showing that the proposed multiplier has a lower delay than a previously designed multiplier at the same number of bits. With respect to the area in Table V, the proposed multiplier is fewer hardware resources by 13% as compared to Ref. [7].

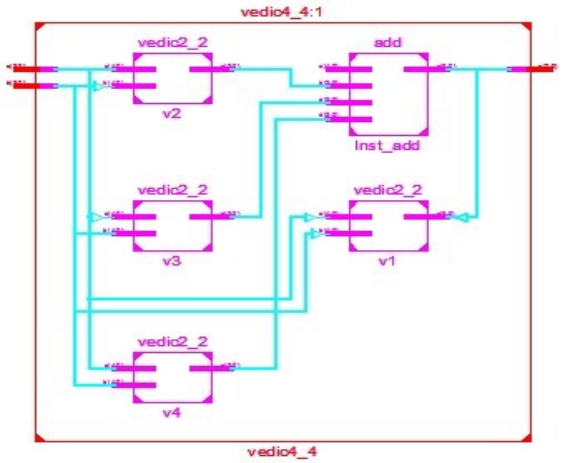

Fig. 7. The RTL schematic for 4-bit proposed multilier.

```
Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                    0 out of 301,440    0%
  Number of Slice LUTs:                     7,243 out of 150,720    4%
    Number used as logic:                   7,222 out of 150,720    4%
      Number using O6 output only:          5,398
      Number using O5 output only:            191
      Number using O5 and O6:               1,633
      Number used as ROM:                       0
    Number used as Memory:                      0 out of  58,400    0%
    Number used exclusively as route-thrus:    21
      Number with same-slice register load:     0
      Number with same-slice carry load:       21
      Number with other load:                   0

Slice Logic Distribution:
  Number of occupied Slices:                2,879 out of  37,680    7%
  Number of LUT Flip Flop pairs used:       7,243
    Number with an unused Flip Flop:        7,243 out of   7,243  100%
    Number with an unused LUT:                  0 out of   7,243    0%
    Number of fully used LUT-FF pairs:          0 out of   7,243    0%
    Number of slice register sites lost
      to control set restrictions:              0 out of 301,440    0%
```

Fig. 8. Map report for 64-bit Proposed design

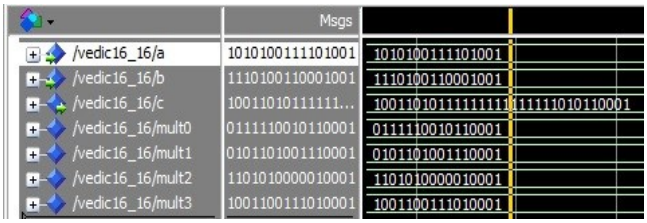

Fig. 9.a. Simulation result for 16-bit Proposed multiplier

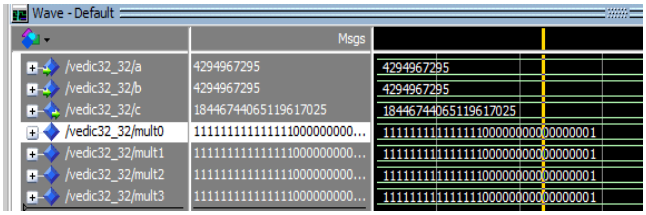

Fig. 9.b. Simulation result for 32-bit proposed Vedic

REFERENCES

[1] A. Jais and P. Palsodkar, "Design and implementation of 64 bit multiplier using vedic algorithm, on Communication and Signal Processing," in Melmaruvathur, India, 2016.

[2] R. A. Javali, R. J. Nayak, A. M. Mhetar, and M. C. Lakkannavar, " Design of high speed carry save adder using carry lookahead adder" in Proceedings of Intern. Conf. on Circuits, Communication, Control and Computing, Bangalore, India, 2014.

[3] P. Verma, "Design of 4x4 bit vedic multiplier using EDA tool," Inter. Journal of Computer Applications , vol. 48, June 2012.

[4] S. Roy, M. Choudhury, R. Puri, and D. Z. Pan, "Polynomial time algorithm for area and power efficient adder synthesis in high-performance designs," IEEE Trans. on Computer-Aided Design of Integ. Circuits and Systems, vol. 35, May 2016.

[5] M. S. A. Menon and R. J. Renjith, "Implementation of 24 bit high speed floating point vedic multiplier," in International Conference on Networks & Advances in Computational Technologies, Trivandrum, India, July 2017|.

[6] S. Arish and R. K. Sharma, "An efficient binary multiplier design for high speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm," in Global Conf. on Communication Tech., Thuckalay, India, April 2015.

[7] S. Z. H. Naqvi, "Design and simulation of enhanced 64-bit vedic multiplier" in inter. Conf. on Applied Electrical Engineering and Computing Tech., Aqaba, Jordan, oct 2017

[8] A. C, A. Chalil, " Performance analysis of montgomery multiplier" Inter. Confer. on Communication and Electronics Systems (ICCES 2017), coimbatore, India

[9] S. G. Mohamed, M. A. Alshewimy, " Design and implementation of new delayy efficient/configurable multiplier using FPGA," in inter. Confer. on Computer Engineering and Systems (ICCES), Cairo, Egypt, Dec. 2017.

[10] G. C. Ram, Y. R. Lakshmanna D. S. Rani, K. B. Sindhuri,"Area efficient modified vedic multiplier," Inter. Conf. on Circuit, Power and Computing Tech. [ICCPCT], Nagercoil, India,2016.

[11] D. K. Kahar, H. Mehta, "High speed vedic multiplier used vedic Mathematics," Inter. Conf. on Intelligent Computing and Control Systems, Madurai, India,2017.