2018 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)

FPGA Implementation of Square and Cube Architecture using Vedic Mathematics

Sampada Barve[1], Sithara Raveendran[2], Charudatta Korde[3], Trilochan Panigrahi[4], Nithin Kumar Y. B.[5], Vasantha M. H.[6]
National Institute of Technology Goa, Goa, India [1,2,3,4,5,6]
Email: (sampadabarve12[1], sithararaveendrann[2], 100494c[3])@gmail.com,
(tpanigrahi[4], nithin.shastri[5], vasanthmh[6])@nitgoa.ac.in

***Abstract*—Squaring and cubing units have importance in various applications in digital signal processing. This paper proposes new squaring architectures based on vedic mathematics sutra of Antyayordashakepi and Dwandwa Yoga (Duplex). Further a new cube architecture based on proposed square and Anurupya sutra is also proposed. The squaring units were implemented for bit size of 8 and 16 while cube was implemented for bit size of 8 on kintex 7 FPGA board. Proposed squarers provided with power delay product of 106.99 and 45.65 whereas cube had power delay product of 444.47 for 8 bits of input.**

Keywords- Vedic maths, Ekadhikena Purvena, Antyayordashakepi, Duplex, Square, Cube, FPGA

I. Introduction

Squaring and Cubing of binary numbers are important arithmetic operations widely used. Squaring operation is useful in digital signal processing (DSP) applications which include image compression, least mean squaring, demodulation, decoding and adaptive filtering[1]. It is also useful in implementing algorithms like Chirp Z-Transform, Fast Fourier Transform etc. Square and cube also have applications in graphic processor, cryptography, pattern recognition and rectangular to polar conversions in various circuits where complete precision result is not needed[1].

Generally, the squaring and cubing functions are implemented using the existing or conventional multiplier techniques rather than use of dedicated units for the same. For calculating the square of a binary number, fast multiplication algorithms like Booths algorithm[2], Wallace trees[3] and Dadda multiplier[4] have been proposed. But multiplication operation requires repetitive addition operations. Again the binary addition is considered as one of the most time-consuming operation in DSP processor[1]. Thus using dedicated squarers and cubes can help to improve overall area and performance of different applications.

The implementation of squaring algorithm takes the advantage that both operands are same. Thus, the number of partial products required can be reduced by eliminating redundant bits which results in a circuit that requires low power and less hardware with reduced delay. As the demand for faster processes increases, special attention need to be given to square and cube functions.

Ekadhikena Purvena Sutra [5] is generalized for finding square of a number. A digit serial algorithm was proposed, which generates 2 digits of final square in every iteration starting from least significant bit. Thus it requires 'n' iterations to evaluate the square of n digit number. Here the number of bits to be processed in every iteration or number of bits per digit can be varied. Thus area and latency product can be varied. Duplex property is used in [6], in which the operand is partitioned into groups of bits and squaring is done by multiplying these sub partitions using duplex property and interlaced multiplication. Also, cubing architecture based on Anurupya Sutra is proposed[6]. Both architectures showed improved delay. This dedicated squarer was compared with the conventional multiplier designed using Radix-4 Modified Booth encoding scheme.

A new squaring architecture is proposed in [7] by considering both operands as same for square. Thus, the total number of partial products required reduces which in turn reduces the total number of elements to be added in a column, i.e. depth. This architecture showed significant improvement in power consumption. In [1], Yavadunam Sutra is used to evaluate square and cube of number. In [8], duplex property was used to calculate square followed by anurupya sutra to find cube. This paper proposes two squaring architectures based on Antyayordashakepi sutra and Duplex method. A new cubing architecture based on proposed square and Anurupya sutra is proposed.

The paper is organised as follows: Section II describes the proposed architectures for square. Section III explains the proposed cube using previously proposed square. Section IV gives implementation details with comparison of simulated results for the proposed architecture. Finally, conclusion are drawn in Section V.

II. Proposed Squaring Algorithm

A. Antyayordashakepi Square

This subsutra states that the rule applied in Ekadhikena Purvena which is used to find square is also applicable to multiplication of two numbers whose least significant digits add to 10. If two numbers are identical except for the least significant digit such that the least significant digits add to 10 then the multiplication of two numbers can be obtained by using this method. The final product is splitted into two parts namely, right partial product and left partial product. Right partial product can be obtained by multiplying Least

0-7695-6618-9/18/$31.00 ©2018 IEEE
DOI 10.1109/iSES.2018.00012

Significant Digit (LSD) of multiplicands. To obtain left partial product, the number formed by removing least significant digit has to be multiplied to a number greater than it by one. Thus the final answer is the concatenation of the two partial products.

Mathematically, for 2 digit Decimal number, $d_0 + \bar{d_0} = 10$, where $d_1, d_0, \bar{d_0}$ each represent a decimal digit,

$$\begin{aligned} d_1d_0 * d_1\bar{d_0} =& (d_10 + d_0) * (d_10 + \bar{d_0}) \\ =& d_10(d_10 + d_0 + \bar{d_0}) + d_0\bar{d_0} \\ =& d_10(d_10 + d_0 + \bar{d_0}) + d_0\bar{d_0} \\ =& d_1(d_1 + 1)00 + d_0\bar{d_0} \\ =& \{d_1(d_1 + 1)\}\{d_0\bar{d_0}\} \end{aligned} \tag{1}$$

Example of above sutra for decimal number:

$$\begin{aligned} \text{Decimal number} &= 23 * 27 \\ 3 + 7 &= 10 \rightarrow Base \\ RPR &\implies 7 * 3 = 21 \\ LPR &\implies 2 * 3 = 6 \\ Ans &= 621 \end{aligned}$$

This rule can be extended for binary number system to calculate square of given number. Since in squaring operation N bit multiplier and multiplicand are identical, all bits and thus (N-1) Most Significant Bits (MSBs) will be identical. Thus, left partial product can be obtained by incrementing the (N-1) bit number by 1 and multiplying it to original (N-1) bit number. If the Least Significant Bit (LSB) is one then the product of LSBs gives 2 bits binary 01, which is right partial product. If the least significant bit is zero then right partial product will be 2 bits binary 00. The final result can be obtained by concatenating the right and left partial product. For 2 digit Binary number Square, b_1b_0, where b_1 and b_0 represents the bits,

$$\begin{aligned} \{b_1b_0\}^2 =& (b_10 + b_0) * (b_10 + b_0) \\ =& b_10(b_10 + b_0 + b_0) + b_0 \\ =& b_10(b_10 + b_00) + b_0 \\ =& b_1(b_1 + b_0)00 + b_0 \\ =& \{b_1 + b_1b_0\}\{0b_0\} \end{aligned}$$

For 3 digit Binary number, $b_2b_1b_0$ where b_2, b_1 and b_0 represents the bits,

$$\begin{aligned} \{b_2b_1b_0\}^2 =& (b_2b_10 + b_0) * (b_2b_10 + b_0) \\ =& b_2b_10(b_2b_10 + b_0 + b_0) + b_0 \\ =& b_2b_1(b_2b_1 + b_0)00 + b_0 \\ =& \{\{b_2b_1\}^2 + (b_2b_1)b_0\}\{0b_0\}. \end{aligned}$$

$$\begin{aligned} \text{Binary Number} &= 1001^2 \\ RPR &\implies 1 * 1 = 01 \end{aligned} \tag{2}$$

$$\begin{aligned} LPR &= 100 * 101 \\ LPR &= 010100 \\ Ans &= \{010100, 01\} \\ Ans &= 01010001 \end{aligned}$$

Thus, in this technique the bit size of partial product to be added varies. From equations it can be seen that the right partial product consists of 2 binary bits with least significant bit being identical to the least significant bit of operand. So no additional computation is required. The left partial product is given by sum of square of most significant N-1 bits of operand with the logical AND product of least significant bit with this same N-1 bit number. So it is possible to use this squaring technique recursively to evaluate the N-1 bit square. This technique can be used to find square of number in left to right style, that is, most significant bits of final result are obtained prior to least significant bits. Thus, it can also be used for calculating approximate square.

B. Duplex Square

Dwandwa Yoga or Duplex method is another vedic maths technique to find square. In this method duplex is taken in two ways i.e. by squaring or by cross multiplication. Here initially duplex of single least significant digit is found followed by finding duplex of two least significant digits and procedure is continued till duplex of full n digit operand is found. Then duplex of all (n-1) most significant digits excluding one least significant digit is calculated and same is repeated till most significant digit is reached. All duplexes are then added after appropriate shifting in proper order taking first duplex as reference. This process is shown below for two digit decimal number (b_1b_0):

$$\begin{aligned} (b_1b_0)^2 & \\ D(b_0) &\implies b_0^2 = b_0 \\ D(b_1b_0) &\implies 2 * b_1 * b_0 \\ D(b_1) &\implies (b_1^2) = b_1 \\ (b_1b_0)^2 &= D(b_1) + D(b_1b_0) + D(b_0) \\ (b_1b_0)^2 &= b_1 + 2 * b_1 * b_0 + b_0 \end{aligned}$$

In the second proposed square, the N digit operand is considered to have two digits with each digit made up of $\frac{N}{2}$ bits. Then using above formula it is possible to evaluate square. The two internal squaring operations are performed using antyayordashakepi sutra and a multiplication is done using conventional $\frac{N}{2} * \frac{N}{2}$ wallace multiplier. The internal squaring operations and a multiplication can be performed simultaneously followed by adding appropriately.

III. Proposed Cube

A. Anurupya Sutra

This is the subsutra of Ekadhikena Purvena. In this, the operand is divided into two parts or digits. Each digit contains some bits of the number. Cube of each digit is found. Also square of a digit, is multiplied to other digit. This product is

added to twice the same number to obtain thrice of this same number. All these are added appropriately. Mathematically, this sutra is given by
$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$
where a and b represent digits of number ab. Example for binary number $(1110)^3$
$= 11^3 + 3 * 11^2 * 10 + 3 * 11 * 10^2 + 10^3$
$= 11011 + 110110 + 100100 + 1000$
$= 11011000000 + 1101100000 + 10010000 + 1000$
$= 101010111000$
Internal cube terms are evaluated using a squaring operation followed by a multiplication operation. All the internal squaring operations are performed using antyayordashakepi square. Multiplication is performed using wallace multiplier. Twice of this product is obtained by left shifting the product of this multiplication by 1 bit. This result is then added to same multiplication product to obtain $3a^2b$ and $3b^2a$.

IV. Implementation

Proposed square I (Antyayordashakepi Square), proposed square II were coded in Verilog. Proposed Square 2 was implemented by combination of Antyayordashakepi and Duplex sutra along with use of wallace multiplication. For this,two internal squarings of bit size 4 and 8 bits each were performed using Antyayordashakepi Square for operand size of 8 and 16 bits respectively. Similarly, a single multiplication using wallace method was performed for bit size of 4 and 8 each. It was compared with the wallace multiplier which is a purely combinational circuit. Square in [5] was coded using finite state machine and sequential in nature. Square in [8] is based on Duplex method and was implemented using combinational circuit. All squarers required were designed, simulated and implemented for unsigned binary numbers with bit size of 8 and 16 on Kintex 7 FPGA. For the comparison, Wallace multiplier of bit size 8 and 16 was designed, simulated and implemented on Kintex 7 FPGA.
The comparison table for the same is shown below.

TABLE I
Utilization report for 8 Bit Square

Parameters	Slice LUTs	Total Power (mW)	Delay(nS)	Power Delay Product(pWs)
Sq[5]	126	1.165	153.435	178.75
Sq[8]	94	10.689	11.925	127.46
Proposed I	46	9.727	11	106.997
Proposed II	33	7.267	6.282	45.651
Wallace	187	15.139	13.521	204.69

It can be seen that all the dedicated squaring circuits showed improved PDP than conventional wallace multiplier for input bit size of 8. Proposed square II showed 53.53% and 83.83% decrease in delay and number of LUTs required. Square[5] exhibited significant percentage decrease in power of 92.30% but had maximum delay out of all. Square[8] and Proposed square I had comparable power and delay but later has better PDP.

TABLE II
Utilization report for 16 Bit Square

Parameters	Slice LUTs	Total Power (mW)	Delay(nS)	Power Delay Product(pWs)
Sq[5]	202	2.653	1302.136	801.566
Sq[8]	193	27.325	17.715	571.81
Proposed I	250	31.884	216.466	525
Proposed II	261	29.545	12.778	377.52
Wallace	833	45.182	17.706	799.99

For 16 bit input, Square[5] used least power with maximum delay. Area reduction in terms of LUTs was provided by all squarer in comparison with wallace multiplier with Square[8] showing 75% reduction in LUTs. Proposed Square I required more LUTs and delay was comparable to that of Square[8]. But Proposed Square II showed 51.7% reduced delay than conventional wallace multiplier and provided with minimum PDP.

TABLE III
Utilization report for 8 Bit Cube

Parameters	Slice LUTs	Total Power (mW)	Delay(nS)	Power Delay Product(pWs)
Anpcube	444	28.661	15.508	444.47
Wallacecube	558	37.167	42.865	1711.98

For Anp cube squaring operation was implemented using Antyayordashakepi square. For wallace cube squaring operation was performed using 8 bit wallace multiplier. In both cases, the second multiplication for cube was performed using 16*8 wallace multiplier. But performance was improved due to reduced number of partial products in square which in turn required lesser number of half and full adders. It was seen that area in terms of LUTs, delay and power was reduced in proposed cube. Delay and PDP reduced significantly by about 60% and 70% respectively.

Bar graphs are shown in Fig. 1, Fig. 2, Fig. 3 and Fig. 4 for variation of Area in terms of number of LUTs, Power, Delay and PDP for input bit size of 8 and 16.

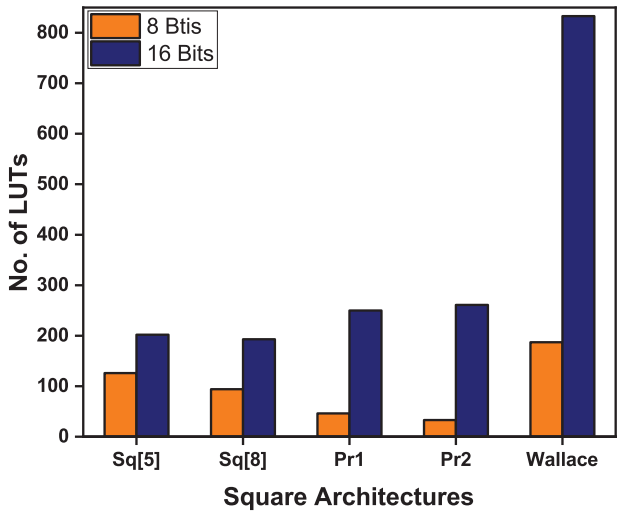

Fig. 1. Area Comparison of Conventional Architectures with Proposed architectures

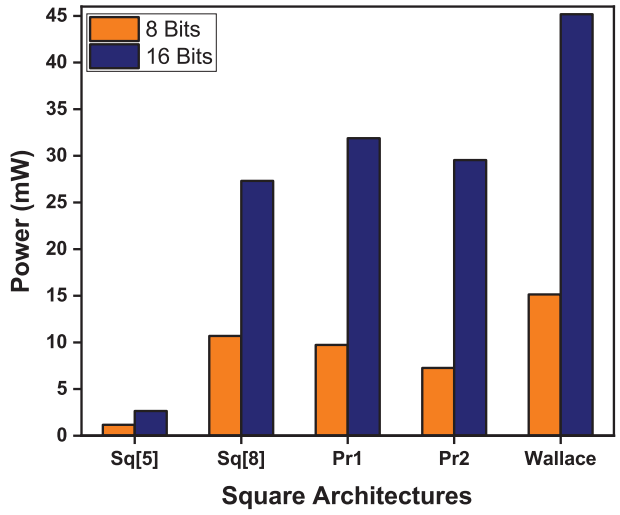

Fig. 2. Power Comparison of Conventional Architectures with Proposed architectures

Number of LUTs required increases significantly for proposed square II and I by 7 and 5 times and for wallace it increases by 4 times but total LUTs required are low for both than that for wallace. Increase in LUTs is least for square [5] where it increases by factor of 1.6 and for square [8] where it increases by factor of 2. With increase in input size, power increment is about two to three times for all except proposed square 2 that requires significant power of around four times than required for 8 bits.

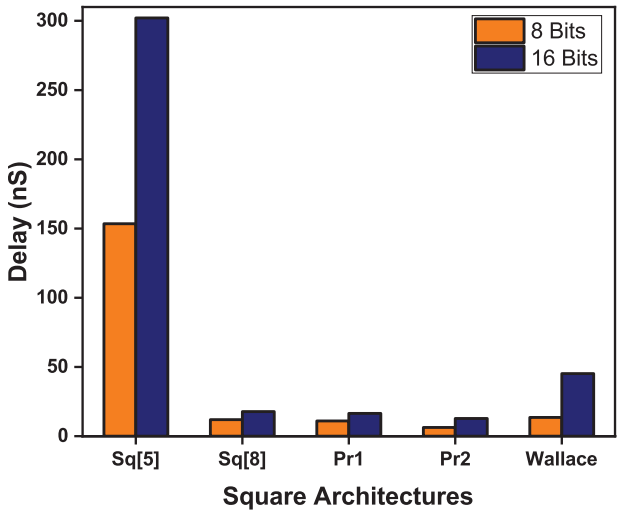

Fig. 3. Delay Comparison of Conventional Architectures with Proposed architectures

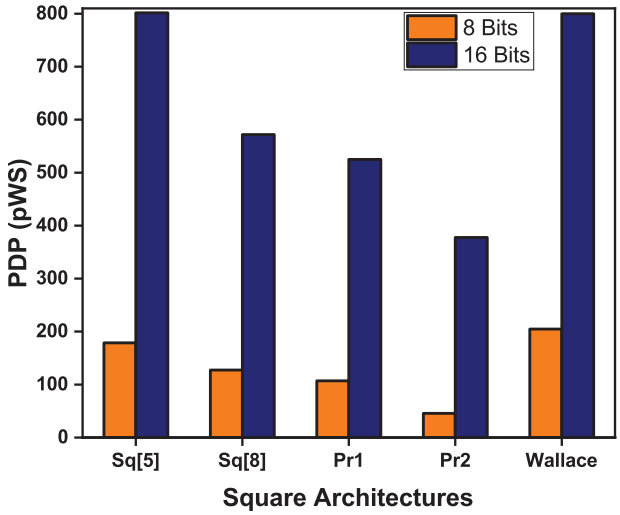

Fig. 4. PDP comparison of Conventional Architectures with Proposed architectures

For all architectures PDP increases significantly with minimum increase of four times. Proposed square II provides around 50% of improvement in PDP than Wallace multiplier for both 8 and 16 bit sizes. Square[8] and Proposed square I also shows better PDP than wallace but its value degrades with increasing bit size. For Square [5] the Delay increases significantly thus affecting the PDP.

V. Conclusion

Proposed squarers showed improved performance in terms of area, power consumption and delay than conventional wallace multiplier. Proposed square II exhibited significant reduction in PDP by 77% and 52% for 8 and 16 bits input respectively

while Antyayordashakepi square provided with PDP improvement of 47% and 34% respectively in comparison with wallace multiplier. Cubing unit using proposed square I showed delay reduction of around 60% and PDP improvement of 74% in comparison with wallace multiplier. Antyayordashakepi Square can be used as MSB to LSB or left to right bit serial squarer. Thus, the proposed vedic squarers are efficient and can be used as dedicated architecture in square intensive applications.

ACKNOWLEDGMENT

This publication is an outcome of the R&D work undertaken in the project under SMDP-C2SD, Department of Electronics and Information Technology, Ministry of Communication & IT, Government of India.

REFERENCES

[1] Ranjan Kumar Barik,Manoranjan Pradhan, "Efficient ASIC and FPGA implementation of cube architecture," IET Computers & Digital Techniques, vol. 11, pp. 43-49, 2017.
[2] A. D. Booth,"A signed binary multiplication technique," Quart. J. Mech. Appl. Math., vol. 4, 1951
[3] C.S.Wallace., "Suggestion for a Fast Multiplier",IEEE Trans. on Computers, vol. 13, pp. 14-17,1964.
[4] L. Dadda,"Squarers for binary numbers in serial form, in Proc. IEEE 7th Symp. Comput. Arithmetic, 1985, pp. 173180.
[5] Saurabh D. Gupta and Mitchell A. Thornton, "A Fixed-Point Squaring Algorithm Using an Implicit Arbitrary Radix Number System," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 6, no. 1, pp. 34-43, 2016.
[6] Himanshu Thapliyal, Saurabh Kotiyal and M.B Srinivas, "Design and Analysis of a Novel Parallel Square and Cube Architecture Based On Ancient Indian Vedic Mathematics," 48th Midwest Symposium on Circuits and Systems, Vol. 2, pp. 1462 1465, 2005
[7] Deshpande A., Draper J., "Squaring units and a comparison with multipliers" Proc. of 53th Int. Symp. on Circuits and Systems, Aug. 2010, pp. 12661269.
[8] Parepalli Ramanammma, "Low Power Square and Cube Architectures using Vedic Sutras," International Journal of Engineering Research and General Science, Volume 5, Issue 3, May-June, 2017.
[9] Maharaja, J.S.S.B.K.T., Vedic mathematics, Motilal Banarsidas Publishers Pvt. Ltd, Delhi, 2009
[10] P. Kornerup and D. W. Matula, Cambridge, Finite Precision Number Systems and Arithmetic, U.K.: Cambridge Univ. Press, 2010.
[11] J. Pihl and E. J. Aas, "A multiplier and squarer generator for high performance DSP applications," Proc. IEEE 39th Midwest Symp. Circuits Syst., vol. 1, pp. 109112, Aug. 1996.
[12] M. Nisha Angeline, S. Valarmathy "Implementation of N-Bit Binary Multiplication Using N-1 Bit Multiplication Based on Nikhilam Sutra and Karatsuba Principles Using Complement Method," Scientific Research Publishing, Circuits and Systems, 2016, 7, 2332-2338
[13] Whitney J. Townsend, Earl E. Swartlander Jr, Jacob J. Abraham,"A Comparison of Dadda and Wallace Multiplier Delays," Proc. SPIE, Advanced Signal Processing Algorithms, Architectures, and Implementations XIII, pp. 552-560, 2003.
[14] Manoranjan Pradhan and Rutuparna Panda,"High speed multiplier using Nikhilam Sutra algorithm of Vedic mathematics," Int. J. Electron, 2014, vol. 101, No. 3, pp. 300307
[15] G.Ganesh Kumar, V.Charishma,"Design of High Speed Vedic Multiplier using Vedic Mathematics Techniques," International Journal of Scientific and Research Publications, Volume 2, Issue 3, March 2012
[16] Swapnil Manohar Mehkarkar, Snehal J. Banarase, "Implementation Of High Speed FIR Filter Based On Ancient Vedic Multiplication Technique," International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 5, May 2014