Design and Implementation of High efficient Two-variable KCM Multiplier

Ravikumar M G[1], Meenakshi.L.Rathod[2]

***Abstract*— In this paper, a high efficient Two-variable Constant Co-efficient Multiplier (KCM) is proposed. The multiplier is designed using Squarers which are designed using Vedic mathematics. This paper focuses on design, implementation and optimization of two- variable Constant Co-efficient Multiplier at algorithmic level. The proposed multiplier design is faster and much efficient than the existing system. The proposed architecture is implemented based on the two-variable Constant Co-efficient Multiplier algorithm to provide low power, area and high speed by design constraint trade-offs. The proposed design is designed and synthesized under ASIC domain using Cadence and functionally verified using Cadence RC Encounter RTL compiler tool. The proposed architecture consumes 23% less power than the existing architecture.**

***Index Terms*— Two variable KCM, Vedic Squarer Algorithm, ROM based logic design, ASIC design**

I. INTRODUCTION

Multiplier is one of the key hardware blocks in most digital systems. The multiplication is performed by "shift and add" method. This includes partial product generation, reduction and addition. Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. As the performance is limited by propagation delay, many multiplier architectures were proposed to minimize the delay. The two-variable constant co-efficient multiplication is one of them.

II. EXISTING TWO-VARIABLE KCM MULTIPLIER

The existing two-variable KCM multiplier's architecture [1] is similar to that of a Constant Coefficient Multiplier (KCM). However, for KCM one input is to be fixed, while the two-variable KCM multiplier can multiply two variables. Multiplication in either a general purpose processor (GPPs) or in custom hardware implementation is generally an expensive operation in terms of latency or hardware area. If one of the operands of a multiplication is known beforehand, the expensive cost of multiplication can be reduced greatly.

In the two-variable KCM, both operands are can be variables. The method is based on ROM approach [1, 2 and 14]. In this a ROM is used for storing the squares of numbers as compared to KCM where the multiples are stored. Thus the amount of ROM used for storing the values is reduced, as it will use for only storing the values of squares.

Method: To find ($a \times b$), first we have to find whether the difference between '*a*' and '*b*' is odd or even. Based on the difference, the product is calculated using (1) and (2) [1].

I. In case of Even Difference
Result of Multiplication = $[\text{Average}]^2 - [\text{Deviation}]^2$…. (1)

II. In case of Odd Difference
Result of Multiplication = $[\text{Average} \times (\text{Average} + 1)] - [\text{Deviation} \times (\text{Deviation}+1)]$…(2)

Where, Average = $[(a+b)/2]$ and Deviation = [Average – smallest (a, b)]

Example 1: $16 \times 12 = 192$

1) Find the difference between (16-12) = 4 an Even Number
2) For Even Difference, Product = $[\text{Average}]^2 - [\text{Deviation}]^2$
   i. Average = $[(a+b)/2]$ = [(16+12)/2] = [28/2] = 14
   ii. Smallest(a,b) = smallest(16,12) =12
   iii. Deviation = Average – Smallest (a,b) = 14 -12 =2
3) Product = 142 - 22= 196 - 4 = 192

Example 2: $15 \times 12 = 180$

1) Find the difference between (15-12) = 3 an Odd Number
2) For Odd Number Difference Product = [Average × (Average + 1)] – [Deviation × (Deviation+1)]
   i. Average = $[(a+b)/2]$ = [(12+15)/2] = 13.5
   ii. Deviation = [Average – smallest(a,b)] = [12.5 – smallest(13,12)] = [13.5 - 12] = 1.5
3) Product = (13×14) - (1×2) = 182 – 2 =180

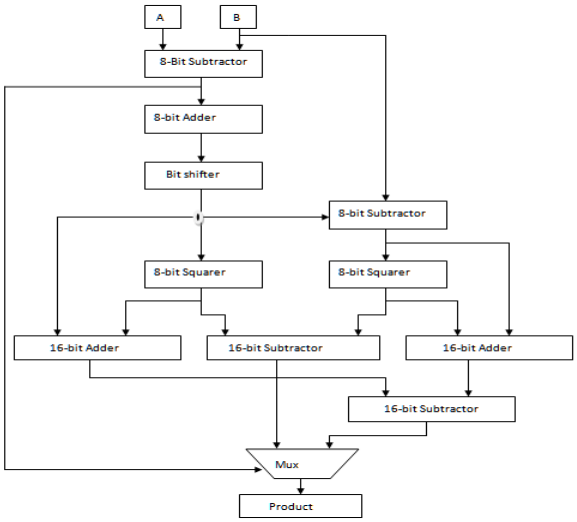


Fig. 1 Two variable KCM (2v-KCM) multiplier using squarers

In the 2v-KCM which is shown in fig. 1, due to high device complexity, the 2v-KCM requires high memory and this leads to high power consumption. Due to this the high heat dissipation takes place and makes device battery life

ISSN: 2278 – 7798

shorter. To avoid high heat dissipation, the cooling costs are high.

These demerits can be stated as,

1. Excess use of adder and subtractor blocks.
2. Separate calculations for even and odd difference between the two operands.
3. Not a power aware and less targeted towards portability.

III. PROPOSED TWO-VARIABLE KCM

To overcome demerits of present 2v-kcm, we are proposing an improved architecture which can withhold its high performance feature and produce low power dissipation. Optimizing the existing 2v-KCM multiplication algorithm, we are proposing an improved 2v-KCM multiplication algorithm as shown below.

According to present 2v-KCM method, the result of multiplication for odd difference is,

Result of Multiplication = [Average × (Average + 1)] – [Deviation x (Deviation + 1)]

Simplifying above equation, we have,

Result of Multiplication = (Average^2 -- Deviation^2) + (Average -- Deviation) …..… (3)

The result of subtraction between Average and Deviation, yields the result, which is equal to b only, therefore,

Result of Multiplication = (Average^2 -- Deviation^2) + b ……..(4)

Thus we can propose a new 2v-KCM method as,

To find ($a \times b$), first we have to find whether the difference between '*a*' and '*b*' is odd or even. Based on the difference, the product is calculated using (3.8) and (3.9).

i. In case of Even Difference
Result of Multiplication= [Average] ^{2}– [Deviation] ^{2} … (5)

ii. In case of Odd Difference
Result of Multiplication = [Average^2 – Deviation^2] - b… (6)

Where, Average = [(a+b)/2] and Deviation = [Average – smallest (a, b)]

The square products of terms Average and Deviation are produced using Vedic Hybrid squarers. Squarers are computing blocks in which same number is multiplied with itself. The Squaring operation is also an essential part of any digital system. These squarers designed using Vedic squarer algorithm [3] and the algorithm is as follows,

1. The given number is divided into two equal numbers of bits. For a n-bit case MSN and LSN will contain n/2 bits each.
2. To find a^2 and b^2 both MSN and LSN are given as input to a ROM which contains the squares of n/2 bit numbers. The square content of MSN is shifted n times towards left to obtain a^2.(Due to the removal of four zeroes)
3. To find $2ab$ first ab is found both a and b are given to an n/2×n/2 array multiplier and the product is shifted n/2+1 times towards left to find $2ab$.
4. The final answer is obtained by adding $2ab$, a^2 and b^2.

The resulting block diagram is,

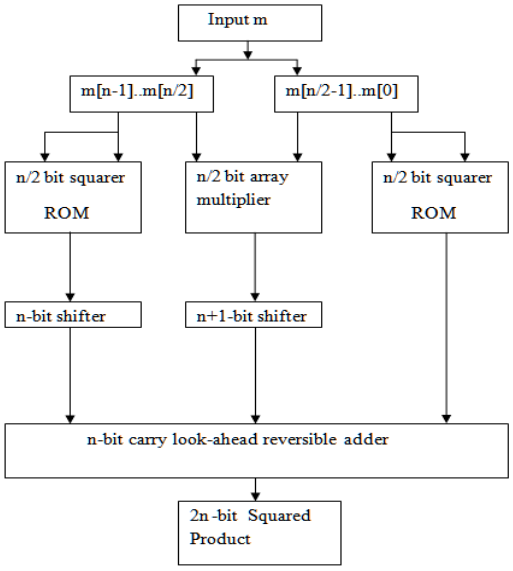


Fig 3.3 Hybrid squarer

Example 4: Find the square of 255.

255_{10}=1111 1111_2

I. Since there are eight digits MSB = 1111_2 LSB = 1111_2

II. MSB^2= 11100001_2 LSB^2 (b^2) = 11100001_2.
To obtain a^2, $11100001\ 00000000_2$. (MSB shifted eight times towards left.)

III. $2ab$, ab = $11100001\ 0000_2$ (4 Bit left shift for the removal of 4 zeroes).
To find $2ab$, 1110000100000_2 (1 Bit shift towards left for multiplication by 2)

IV. $a^2+2ab+b^2$,
$1110000100000000_2+1110000100000_2+11100001_2$ = 1111111000000001_2.

The architecture designed based on our proposed algorithm is shown in Fig. 2. The proposed architecture shows a simplified architecture for a 2v-KCM which is both efficient by providing low power dissipation as well as by performing high performance with 2v-KCM multiplication. The efficiency is achieved by minimizing the area requirement. To obtain multiplication result for two operands with odd difference, only value of second operand is added to the result of multiplication for even difference instead of calculating again the squares of values of the terms Average and Deviation of the two operands. Thus the efficiency is achieved. The mechanism of proposed algorithm is shown in Example 3.

Example 3: 15 × 12 = 180

1) Find the difference between (15-12) = 3 an Odd Number

2) For Odd Number Difference Product = [(Average) ^{2} – (Deviation) ^{2}] -- b

i. Average = [(a+b)/2] = [(12+15)/2] = 13.5

ii. Deviation = [Average – smallest(a,b)] = [12.5 – smallest(13,12)] = [13.5 - 12] = 1.5

ISSN: 2278 – 7798

3) Product = $(13^2 - 1^2) + 12 = (169 - 1) + 12 = 168 + 12 = 180$.

(Neglecting values after the decimal point)

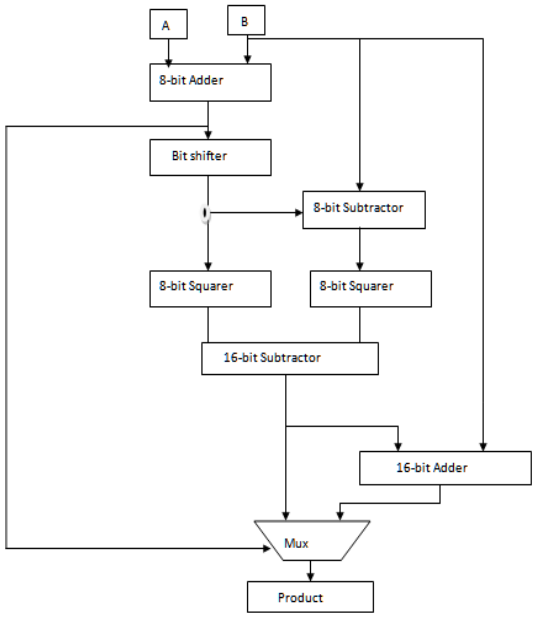


Fig. 2 High efficient 2v-KCM

IV. EXPERIMENTAL RESULTS

The existing and proposed design architectures are implemented for ASIC domain using Cadence tool are shown in Fig.3 and Fig. 4 respectively. The Cadence provides a well-balanced logic structure which isolates critical paths, reduces power, area, and congestion in off-critical logic and enables faster timing closure and design convergence through placement and routing.

The following results obtained by Cadence RC Encounter compiler.

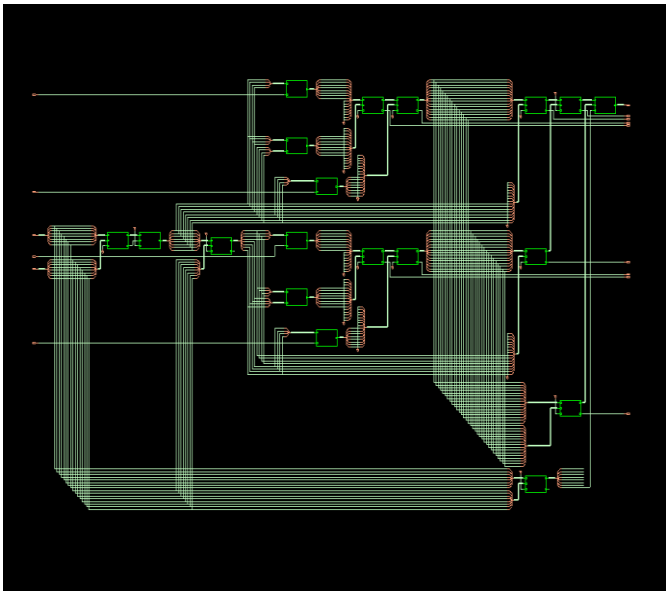


Fig. 3 The RTL view of existing 2v-KCM in Cadence

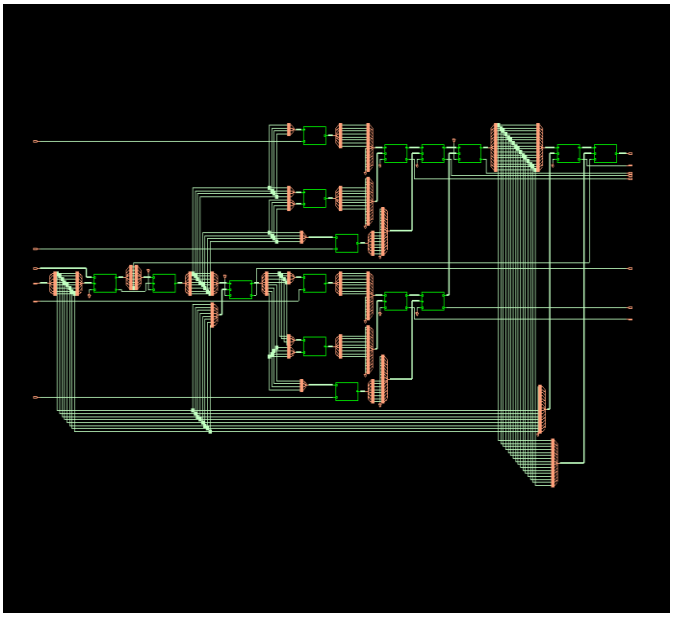


Fig. 4 The RTL view of proposed 2v-KCM in Cadence

TABLE I. COMPARISON OF HARDWARE CELL REQUIREMENTS FOR 8X8 2v-KCM

Gates	Instances		Reduction (%)
	Existing	*Proposed*	
Full adder	16	16	0
Half adder	8	8	0
AND	336	256	23.81
OR	152	112	26.32
XOR	304	224	26.32
TOTAL	816	616	24.51

TABLE II. COMPARISON OF EXISTING AND PROPOSED ARCHITECTURE OF 8X8 2v-KCM WITH THE PARAMETERS AND THEIR IMPROVEMENT

Parameter	Existing	Proposed	Reduction (%)
TIMING (ns)	9012 R	8979 R	0.36
AREA(cells)	1350	1070	20.74
LEAKAGE POWER (nW)	6219.472	5177.192	16.75
DYNAMIC POWER (nW)	223814.611	170628.431	23.76
TOTAL POWER (nW)	230034.083	175805.623	23.57

From the TABLE I and TABLE II, it is inferred that, the proposed 2v-KCM architecture requires less area than the

ISSN: 2278 – 7798

existing architecture and hence produces lesser power dissipation with improved speed.

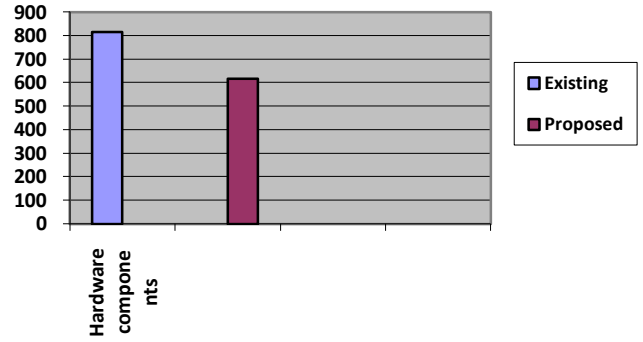


Fig. 5 Hardware components comparison for 8x8

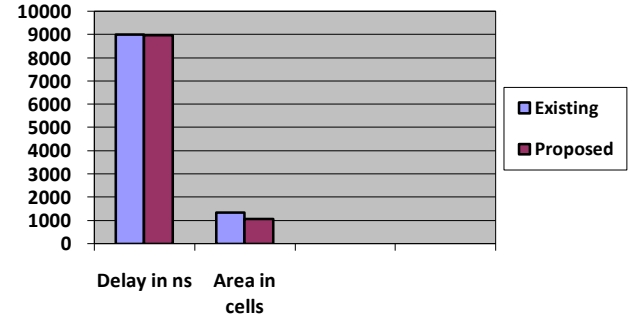


Fig. 6 Speed and Area comparisons for 8x8

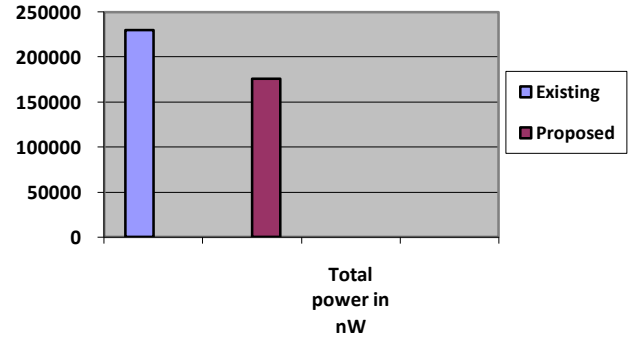


Fig. 7 Power comparison for 8x8

Observations from the results and graph figures:

1. The Power dissipation is reduced as the area reduced.
2. Timing is unaltered with the optimizations and it enables low power with same performance technological corner.
3. Reduced area has reduced the leakage power
4. Dynamic power is reduced due to the lesser transition between the gates

CONCLUSION

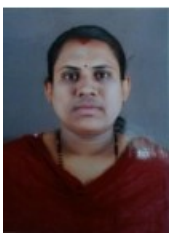
The proposed architecture is designed for low power dissipation. The architecture is designed based on the proposed new 2v-KCM algorithm using squarers which designed using binary squaring algorithm. The proposed architecture results 23.76% of dynamic power, 20.74% of area and 0.36% of time delay reductions. These results are obtained using Cadence tool. The proposed architecture applicable to all computing circuits as the multiplier is the key element. The main attraction of this architecture is the conventional multiplication is replaced by additions and subtractions. The proposed architecture can be optimized for further levels in terms of performance.

REFERENCES

[1] L.Sriraman, T. N. Prabakar, "Design and Implementation of Two-Variable KCM using Multiplier using KCM and Vedic Mathematics" in 1st International Conference on Recent Advancements in Information Technology, 2012.
[2] L.Sriraman, T. N. Prabakar, "FPGA implementation of high performance multiplier using squarer" in Int. Jr. of Advanced Computer Engineering & Architecture Vol. 2 No. 2 (June-December, 2012)
[3] Chandra Mohan Umapathy, "High Speed Squarer using Vedic Mathematics", an open source available in Vedic mathematics Forum
[4] Maii T. Emam, Layle A. A. Elsayed," Reversible Full Adder/Subtractor", 2010 Xl International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD)
[5] John.P.Uyemera, "Introduction to VLSI Circuits and Systems", ISBN13: 9788126509157
[6] W. B. Vasantha Kandasamy, Florentin Smarandache "VEDIC MATHEMATICS - 'VEDIC' OR 'MATHEMATICS': A FUZZY & NEUTROSOPHIC ANALYSIS", 2006
[7] ACG.inc. "Multiplication in FPGAs".
[8] K.-J. Cho and J.-G. Chung , "Parallel squarer design using pre-calculated sums of partial products" , ELECTRONICS LETTERS, 6th December 2007 ,Vol. 43 No. 25.
[9] J Bhasker, "Verilog HDL Synthesis: A practical Primer", Star Galaxy Publishing.
[10] Steven Medina September 18th, 2007, CSC343 Fall 2007.
[11] Peter J Ashenden, "Digital Design an Embedded Systems Approach Using Verilog", Morgan Kaufmann publications, ISBN 978-0-12-369527-7
[12] "ISE Design Flow Overview", Xilinx® Inc.
[13] "Introduction to ASIC Design Methodology" ECE-520/ECE-420 ~ Spring 1999 ~ Rev. 99.1, Dr. Paul Franzon, Scott Perelstein, Amber Hurst
[14] Paul.B.C., Fujita.F.S., Okajima.M., "ROM Based Logic (RBL) Design: A Low-Power 16 Bit Multiplier", IEEE Journal of Solid State Circuits, Volume 44, Issue 11, Pg. 2935-2942, Nov 2009.

PG Student, VLSI design & Embedded Systems, Electronics and Communication Engineering Dept, Dr. Ambedkar Institute of Technology, Bangalore-56, IN.

Paper published: on **"FPGA implementation of High performance Multiplier using Squarers with Vedic mathematics",** 3rd International Conference on Engineering and Technology ICET-2014, May 2014, ISSN: 978-3-643-24819-02.



Asst. Professor, Electronics and Communication Engineering Dept, Dr. Ambedkar Institute of Technology, Bangalore-56, IN.

Specialized in VLSI Design, Embedded Systems, Digital Electronics and Communications

ISSN: 2278 – 7798