DOI: 10.21917/ijme.2017.0071

A 16-BIT HIGH-SPEED MULTIPLIER DESIGN BASED ON KARATSUBA ALGORITHM AND URDHVA-TIRYAGBHYAM THEOREM USING MODIFIED GDI CELLS FOR LOW POWER AND AREA CONSTRAINTS

Bobby Nelson[1] **and Ravi Tiwari**[2]
Department of Electronics and Communication Engineering, Shri Shankaracharya Technical Campus, India

Abstract:

The paper entails the design of a 16-bit multiplier with the combined application of Karatsuba algorithm and the Urdhva-Tiryagbhyam (UT) theorem and the implementation of the multiplier architecture in Modified-Gate-Diffusion-Input (Mod-GDI) cells for improving the area and power constraints in the proposed novel hybrid multiplier.

Keywords:

Area-Efficient, GDI, Karatsuba Algorithm, Multiplier, Urdhva-Tiryagbhyam Theorem

1. INTRODUCTION

Multiplication, being one of the four elementary arithmetic operations, happens to be one of the most crucial processes for an ALU inside a processor. It also finds its use in various digital processing operations of varying bit-widths and thus plays the role of one of the most important data-paths beside the Adder. Binary multiplication however, happens to be one of the most complex operations when it comes to the digital design and implementation and thereby becomes one of the popular fields of research in VLSI.

The multiplication process [13] in its essence, doesn't deviate much in the binary number system from its decimal equivalent, i.e., one of the multiplicands is iteratively multiplied with the individual bits of the other in order to generate the partial product terms, which are then shifted weight-wise and added to yield the final result. The process, seemingly simple and hierarchically implementable, requires extensive hardware in actuality and is marred by a heavy adder delay.

We aim to tackle these shortcomings with a combined approach of the Karatsuba algorithm and the Urdhva-Tiryagbhyam (UT) theorem. These algorithms paired with the implementation of the design using Modified-Gate-Diffusion-Input (Mod-GDI) cells provide the proposed multiplier design an unprecedented reduction in area and power dissipation.

The UT theorem efficiently reduces the complexity of the mathematical process and improves the speed but doesn't exactly offer a considerable reduction in the overall area. The Karatsuba algorithm minimizes the iterative steps in multiplication process but still doesn't simplify the process of multiplication in itself.

The Gate-Diffusion-Input (GDI) cells offer a minimal-area approach for designing the sub-circuits however, they are marred by partial swing problems which render them useless in cascaded architectures. The shortcomings of the two algorithms can be overcome by combining the strengths of both into a novel hybrid design implemented using the Mod-GDI based cells which have been modified to counter the partial swing problem while taking advantage of the minimalism offered by the GDI cells.

The paper has been organized as follows: Section 2 provides a brief survey leading to the current work. Section 3 discusses the proposed methodology for the work. Section 4 discusses the implementation of the proposed design using Tanner EDA Tools. The results of the proposed design have been discussed in section 5. The conclusion has been offered in section 6.

2. LITERATURE SURVEY

A lot many researches have been conducted in the designing of better performing multiplier circuits based on different algorithms [2]-[8].

The design of the hybrid multiplier based on the Karatsuba algorithm and the UT theorem and its implementation in FPGAs has been discussed in [2]. It provides the backbone to our research but doesn't delve deeper into optimization of the used methodology as it is restricted, in terms of the tools used, to only logical synthesis.

The comparison of the UT Multiplier architectures with hierarchical array multipliers has been provided in [3]. The design techniques followed for a multiplier based on the Karatsuba algorithm have been described in [4] and [5].

The design techniques followed for a multiplier based on the UT theorem have been described in [6] and [7]. The use of the GDI technology for various logical circuits has been presented in [8].

The designs do improve the different parameters for all these architectures however, few of the problematic aspects in all these designs are as follows:

- The UT theorem efficiently reduces the complexity of the mathematical process but doesn't exactly offer a considerable reduction in the overall area.
- The Karatsuba algorithm minimizes the iterative steps in multiplication process but still doesn't simplify the process of multiplication in itself.
- The Gate-Diffusion-Input (GDI) cells offer a minimal-area approach for designing the sub-circuits however, they are marred by partial swing problems which render them useless in cascaded architectures.

The two algorithmic drawbacks can be overcome by combining the strengths of both into a novel hybrid design implemented in the Mod-GDI cells which have been modified so as to counter the partial swing problem while taking advantage of the minimalism offered by the GDI cells.

3. METHODOLOGY

3.1 KARATSUBA ALGORITHM

The Karatsuba algorithm [9], [10] is a fast multiplication algorithm which was designed by Anatoly Karatsuba in 1960 and published in 1962. It reduces the multiplication of two n-digit numbers to at most $N\log_2 3$ ($N^{1.59}$ single-digit multiplications in general). It is therefore faster than the classical algorithm, which requires N^2 single-digit products. For example, the Karatsuba algorithm requires 3^{10} = 59,049 single-digit multiplications to multiply two 1024-digit numbers (n = 1024 = 2^{10}), whereas the classical algorithm requires $(2^{10})^2$ = 1,048,576.

Considering two 16-bits wide binary variables, X and Y for the example. The numbers are broken down as follows into the two significant halves:

$$X = X_H \times 2^8 + X_L \tag{1}$$

$$Y = Y_H \times 2^8 + Y_L \tag{2}$$

where, X_H and Y_H represent the two most significant halves of X and Y respectively while X_L and Y_L represent their two least significant halves. Hence,

$$XY = (X_H \times 2^8 + X_L)\times(Y_H \times 2^8 + Y_L) \tag{3}$$

The above expression shows the multiplication of the two binary numerals and when expanded they yield four multiplication terms, each N-bits wide as shown below:

$$A = X_HY_H \tag{4}$$

$$B = X_HY_L + X_L Y_H \tag{5}$$

$$C = X_LY_L \tag{6}$$

$$XY = A\times2^{16} + B\times2^8 + C \tag{7}$$

where, A, B and C represent the three combinatorial expressions. A and C each require one multiplier while B requires two.

So, the long multiplication technique here requires a total of four 8-bit multipliers for proper implementation. However, going by the Karatsuba approach, the multiplication terms shall be reduced by substituting one stage of multiplier with adders and subtractors as follows:

$$B' = (X_H+X_L)\times(Y_H + Y_L) - A - C = B \tag{8}$$

Here, B' represents an alternative manner of representing the same term B with the calculation of just one multiplication term as opposed to the two terms in case of B.

$$XY = A\times2^{16} + B'\times2^8 + C \tag{9}$$

The above expression depicts the numerical representation of the Karatsuba algorithm involving the calculation of three multiplicands as opposed to the four required in case of any classical approach.

Mathematically, the Karatsuba algorithm thus effectively reduces the operational overheads of the multiplier in terms of area and power. However, the algorithm alone doesn't really simplify the design of the multiplier core in the design. It has been used in conjunction with UT theorem for designing the multiplier cores in the proposed hybrid design.

3.2 URDHVA-TIRYAGBHYAM THEOREM

The UT theorem [11], [12] is a useful tool in Vedic Mathematics employed in the multiplication of two numbers in the decimal number system. Even though the UT theorem had been developed for the decimal number system, the theorem appropriately fits the binary number system as well. In the proposed design, we apply the same ideas to the binary number system, in 8-bits, to make the proposed algorithm compatible with the digital hardware. The name of the theorem literally translates to "vertically and crosswise" from Sanskrit. The Fig.2 shows the generation of the term 'G' according to the UT theorem and the rest of the terms are shown in the equation below,

$$\begin{aligned}
D &= A_0B_0\\
E &= A_0B_1 + A_1B_0\\
F &= A_0B_2 + A_1B_1 + A_2B_0\\
G &= A_0B_3 + A_1B_2 + A_2B_1 + A_3B_0\\
H &= A_0B_4 + A_1B_3 + A_2B_2 + A_3B_1 + A_4B_0\\
I &= A_0B_5 + A_1B_4 + A_2B_3 + A_3B_2 + A_4B_1 + A_5B_0\\
J &= A_0B_6 + A_1B_5 + A_2B_4 + A_3B_3 + A_4B_2 + A_5B_1 + A_6B_0\\
K &= A_0B_7 + A_1B_6 + A_2B_5 + A_3B_4 + A_4B_3 + A_5B_2 + A_6B_1 + A_7B_0\\
L &= A_1B_7 + A_2B_6 + A_3B_5 + A_4B_4 + A_5B_3 + A_6B_2 + A_7B_1\\
M &= A_2B_7 + A_3B_6 + A_4B_5 + A_5B_4 + A_6B_3 + A_7B_2\\
N &= A_3B_7 + A_4B_6 + A_5B_5 + A_6B_4 + A_7B_3\\
O &= A_4B_7 + A_5B_6 + A_6B_5 + A_7B_4\\
P &= A_5B_7 + A_6B_6 + A_7B_5\\
Q &= A_6B_7 + A_5B_6\\
R &= A_7B_7
\end{aligned} \tag{10}$$

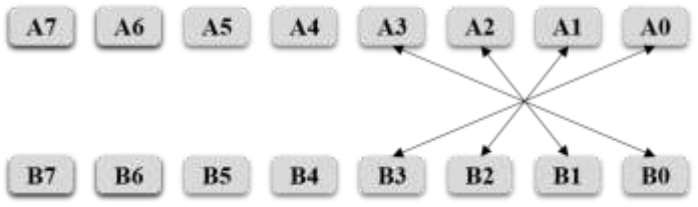


Fig.1. Generation of term 'G'

The implementation of the partial product terms generated thusly can be done by using an array of AND gates and adders. The terms shall have variable bit-widths with 'K' being the widest, with a bit-width of 4 bits.

The Table.1 serves as an easy tool for easily and minimally realizing the combinational logic for the summing of the partial product terms so as to generate the final products. The $C's$ represent the carry terms and actually depict a symbolic collective carry from all the previous stages of addition. Finally, the product is obtained as $X_{16\text{-}bits}$.

However, one important consideration to be taken care of is that the additive product in case of B' in Eq.(8) can have more than 8-bits which would render this multiplier design for 8-bits, useless. Hence, the multiplier that generates the product B' has to process 9-bit multiplicands. The design of this 9-bit multiplier would be exactly like the 8-bit multiplier explained so far but would require a few extra components and would be slightly larger in area. It would not even compare with the addition of a whole extra multiplier though, as in the traditional approach without employing the Karatsuba algorithm.

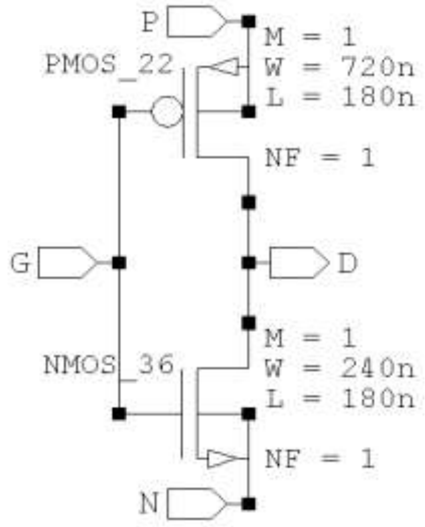
Table.1. Grid representation of partial product addition

															D
												C	E_1	E_0	
										C	C	F_1	F_0		
									C	G_2	G_1	G_0			
								C	H_2	H_1	H_0				
							C	I_2	I_1	I_0					
					C	C	J_2	J_1	J_0						
					K_3	K_2	K_1	K_0							
				C	L_2	L_1	L_0								
			C	M_2	M_1	M_0									
		C	N_2	N_1	N_0										
		O_2	O_1	O_0											
	C	P_1	P_0												
	Q_1	Q_0													
C	R														
X_{15}	X_{14}	X_{13}	X_{12}	X_{11}	X_{10}	X_9	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0

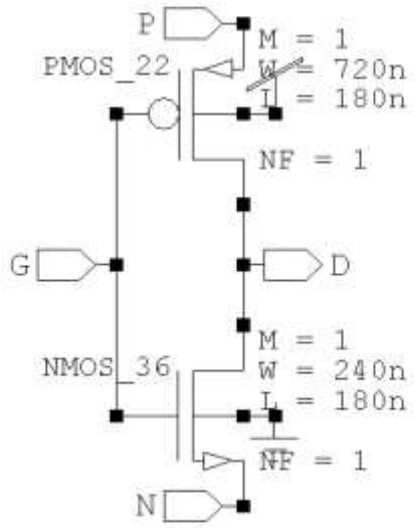
3.3 MODIFIED GDI CELLS

The GDI design technique [1] was introduced as a promising alternative to the CMOS logic design style. GDI methodology allows implementation of a wide range of complex logic functions using merely two transistors.

The Fig.2.(a) shows the basic construction of a GDI cell. The designs have been implemented in 180nm technology with a W_p/W_n ratio of 3, for fairly equal rise and fall times. In conventional GDI cells, the gates of the PMOS and NMOS devices are shorted together to act as an input *G*, the sources terminals are individually shorted with the respective substrates to yield the *P* and *N* terminals.



(a)



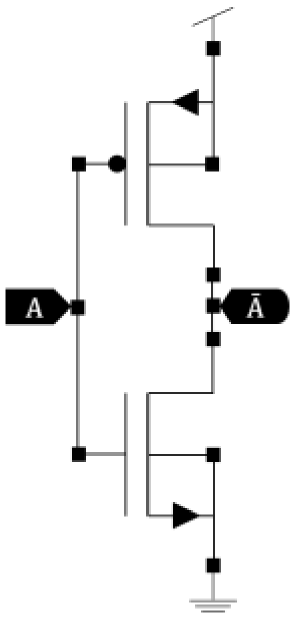
(b)

Fig.2. (a) A GDI cell vs (b) A modified GDI cell

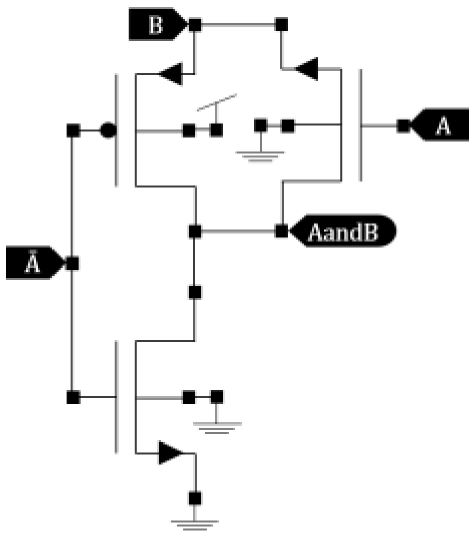
The simple configuration of mere two MOSFETs is capable of producing many complex logic functions. However, there's a certain caveat of partial swing in GDI cells which renders them practically unusable for any cascade connection with other gates. It also leads to wild harmonics in the output signals which dissipate more power than saved. Hence, the actual utility requires a few tiny modifications to the basic GDI cell design as evident in the Fig.2.(b), while the functionality remains the same. The implementation of the basic logic gates using the Mod-GDI cells has been shown in Fig.3.

4. IMPLEMENTATION

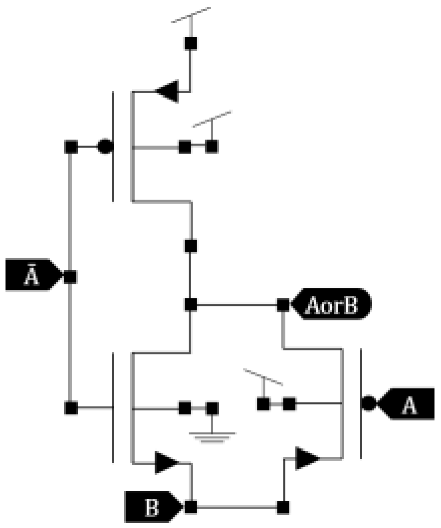
Implementation of the sub-circuits has been carries out using the Mod-GDI cells. Although the Mod-GDI cells do have a better response and lower harmonics than GDI cells but they still have the partial swing problem which needs to be addressed for the individual sub-circuits.



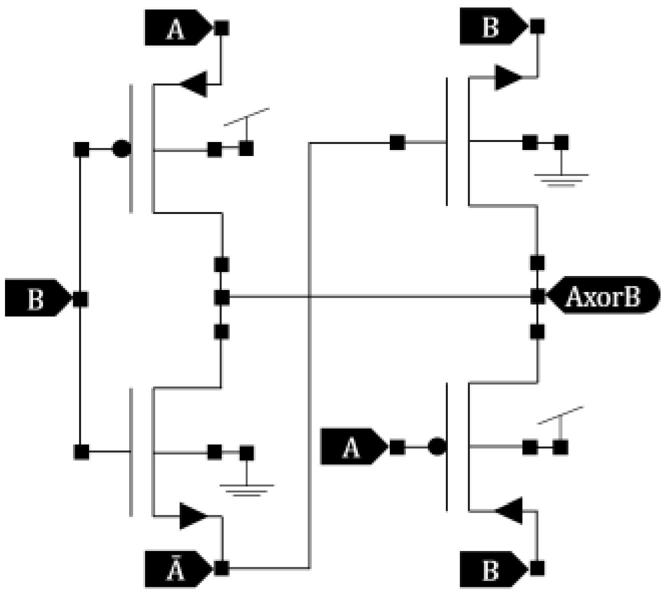
(a)



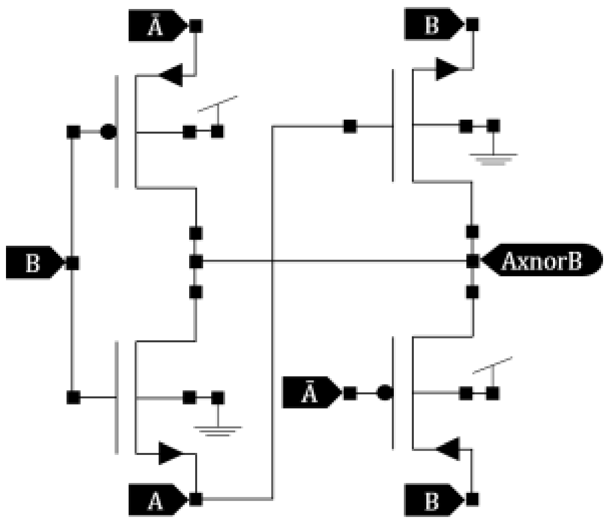
(b)



(c)

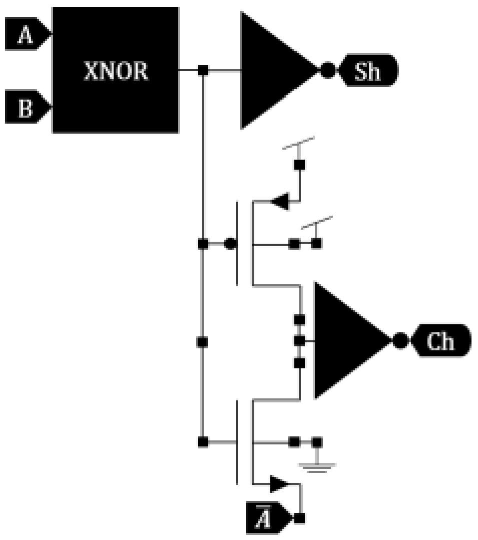


(d)

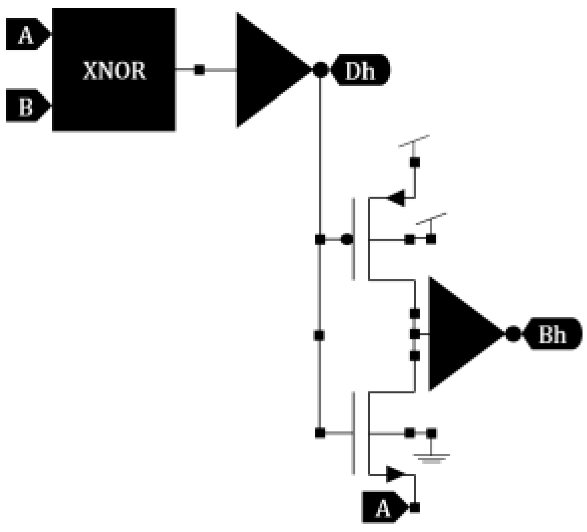


(e)

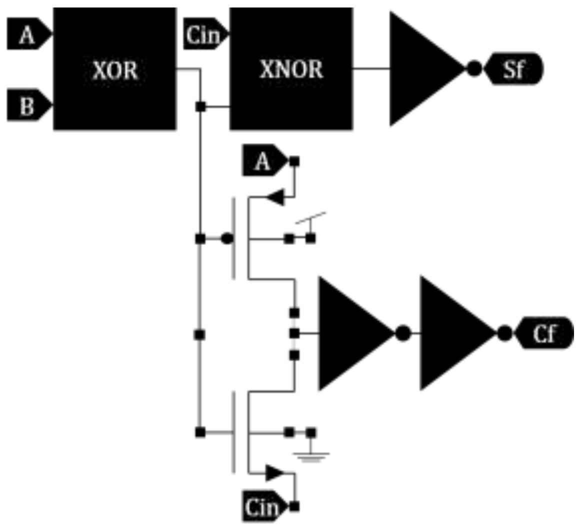
Fig.3. Implementation of basic logic gates in Mod-GDI logic (a) Inverter, (b) AND, (c) OR, (d) XOR, (e) XNOR



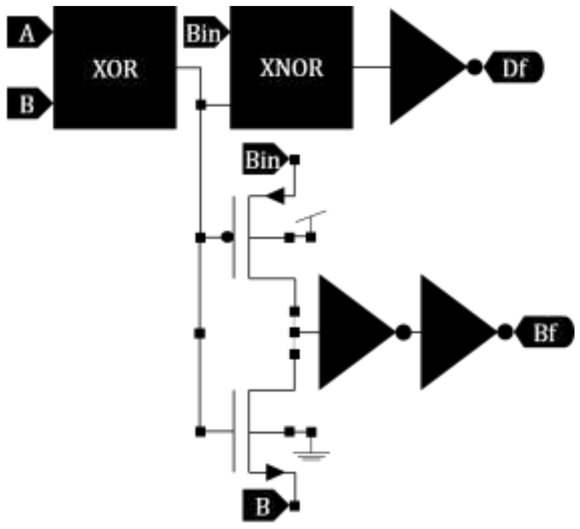
(a)



(b)



(c)



(d)

Fig.4. Implementation of adder and subtractor blocks (a) Half-adder (b) Half-subtractor (c) Full-adder (d) Full-subtractor

The Fig.3 represents the various basic logic gates. The gates have been optimized in Mod-GDI logic for minimum transistor count while maintaining a full-swing output.

The Fig.4 depicts the implementation of the adder and subtractor blocks using the basic gates constructed in Fig.4. The consideration again is to use minimal transistor count for full-swing at output. The Table.2 shows a comparative analysis of transistor count for CMOS vs Mod-GDI.

Table.2. CMOS vs. Mod-GDI Cells

Function	CMOS	Mod-GDI Cells
Inverter	2	2
OR*	6	5
AND	4	3
XOR	12	4
XNOR	12	4
Half-Adder*	22	12
Half-Subtractor*	22	12
Full-Adder*	54	20
Full-Subtractor*	54	20

*Inverters considered inherent in the circuit

The final UT multiplier architecture has been shown in the Fig. 5. It represents the generic UT multiplier architecture for the two kinds of multiplier cores: the 8-bit multiplier core, of which two instances shall be used in the final design to yield the *A* and *C* terms from Eq.(8) while the term *B′* shall be calculated using the 9-bit multiplier core.

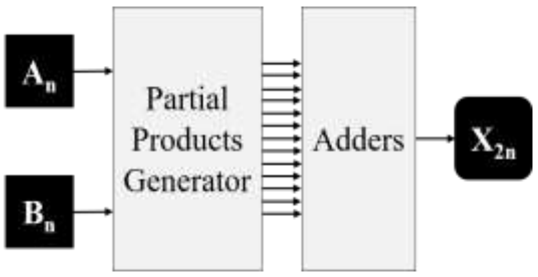


Fig.5. Generic UT Multiplier Architecture

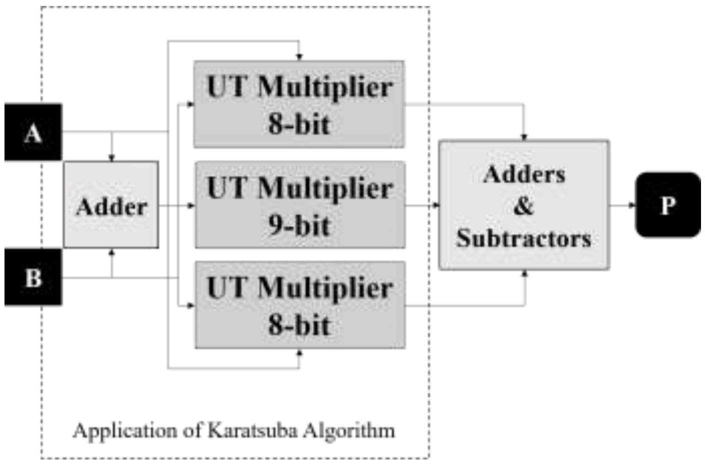


Fig.6. Proposed 16-bit multiplier architecture

Lastly, the results from these multipliers have to be processed as per the Eq.(8), to yield the final product. The basic ripple carry adders have been used in the design for this purpose using the adder and subtractor cells designed in Fig.4. The final proposed multiplier architecture has been shown in the Fig.6.

5. SIMULATION RESULTS AND DISCUSSION

The architecture has been implemented in T-Spice v16.0 at 180nm technology. The Fig.8 shows the simulation output of the proposed 16-bit multiplier for an input combination of all 0's as well as all 1's, with a 50% duty-cycle pulse for all the input signals. The simulation graph signals are highlighted at 1.8V i.e., at the high-state in dark to show the output of the multiplier for the all 1's input combination.

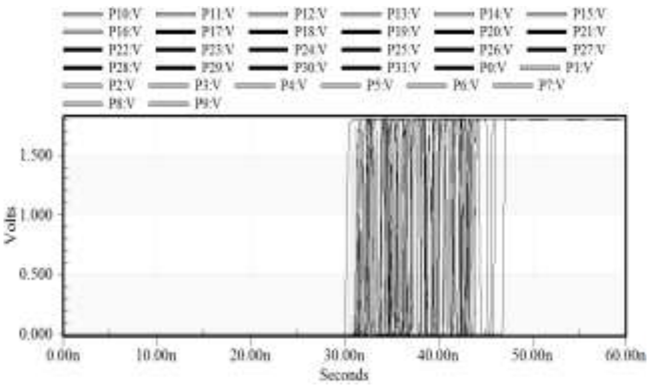


Fig.7. Simulation results for an input combination of all 1's

Table.3. Comparative analysis of multipliers

Multipliers	MOS-FETs	Power mW	Delay ns	PDP pJ	EDP 10^{-21}Js
CMOS Hierarchical Array Multiplier (CHAM) [3]	15776	5.79	24.57	142.28	3495
CMOS UT Multiplier (CUTM) [3]	12864	4.64	22.58	104.77	2366
Mod-GDI K-UT Multiplier (MGKUTM)*	6077	2.98	19.96	59.48	1187

*Proposed multiplier

The proposed multiplier architecture has been tested with numerous pseudorandom inputs for the calculation of the worst-case power and delay values. It has been found to yield conducive results for all the test inputs. The Table.3 presents a comparative analysis of the proposed Mod-GDI Karatsuba-UT Multiplier (MGKUTM) with a CMOS UT Multiplier (CUTM) as well as a CMOS Hierarchical Array Multiplier (CHAM).

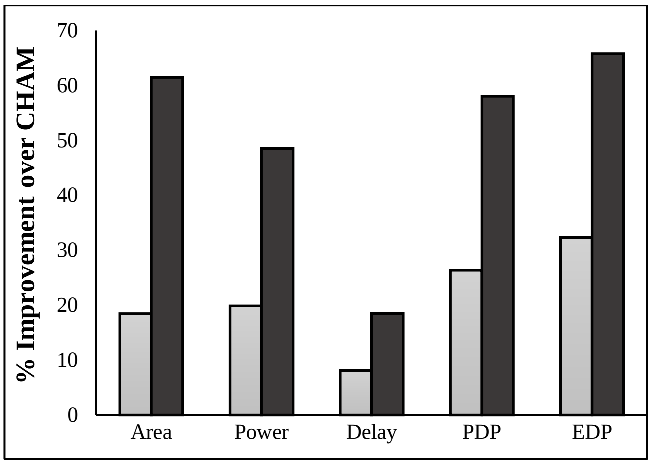


Fig.8. Comparison of percentage improvements in various parameters of CUTM and MGKUTM over CHAM

6. CONCLUSION

From the results obtained in the Table.3 and Fig.8, it has been effectively proved that our proposed multiplier design works better than the other designs in terms of all the parameters in comparison. The most prominent improvement achieved over other designs has been in terms of the area or transistor count. The proposed hybrid multiplier design has thus effectively reduced the area-constraints and also reduced the power consumption of the multiplier unit while the speed of operation has also been marginally improved.

REFERENCES

[1] A. Morgenshtein, A. Fish and I. Wagner, "Gate-Diffusion Input (GDI): A Power-Efficient Method for Digital Combinatorial Circuits", *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 10, No. 5, pp. 566-581, 2002.

[2] S. Arish and R.K. Sharma, "An Efficient Binary Multiplier Design for High Speed Applications using Karatsuba Algorithm and Urdhva-Tiryagbhyam Algorithm", *Proceedings of Global Conference on Communication Technologies*, pp. 192-196, 2015.

[3] Arushi Somani, Dheeraj Jain, Sanjay Jaiswal, Kumkum Verma and Swati Kasht, "Compare Vedic Multipliers with Conventional Hierarchical array of array multiplier", *International Journal of Computer Technology and Electronics Engineering*, Vol. 2, No. 6, pp. 52-55, 2012.

[4] Anand Mehta, C.B. Bidhul, Sajeevan Joseph and P. Jayakrishnan, "Implementation of Single Precision Floating Point Multiplier using Karatsuba Algorithm", *Proceedings of International Conference on Green Computing, Communication and Conservation of Energy*, pp. 254-256, 2013.

[5] C. Eyupoglu, "Performance Analysis of Karatsuba Multiplication Algorithm for Different Bit Lengths", *Procedia-Social and Behavioral Sciences*, Vol. 195, pp. 1860-1864, 2015.

[6] K. Narendra and S. Pandu, "Low Power Area-Efficient Adiabatic Vedic Multiplier", *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, Vol. 3, No. 8, pp. 11027-11032, 2014.

[7] A.M. Kareem and P. Kumar, "VLSI Implementation of High Speed-Low Power-Area Efficient Multiplier Using Modified Vedic Mathematical Techniques", *Recent Patents on Computer Science*, Vol. 9, No. 3, pp. 216-221, 2017.

[8] S. Kaur and B. Singh, "Design and Performance Analysis of Various Adders and Multipliers using GDI Technique", *International Journal of VLSI Design and Communication Systems*, Vol. 6, No. 5, pp. 45-56, 2015.

[9] Paul Zimmermann and Richard P. Brent, "*Modern Computer Arithmetic*", Cambridge University Press, 2011.

[10] Keith O. Geddes, Stephen R. Czapor and George Labahn, "*Algorithms for Computer Algebra*", Springer, 1992.

[11] A.P. Nicholas, K. Williams and J. Pickles, "*Vertically and Crosswise*", Inspiration Books, 2010.

[12] K. Williams and M. Gaskell, "*The Cosmic Computer*", Inspiration Books, 1997.

[13] Neil Weste and David Harris, "*Principles of CMOS VLSI Design*", 4th Edition, Pearson, 2011.