An improved implementation of hierarchy array multiplier using CslA adder and full swing GDI logic

Shoba Mohan and Nakkeeran Rangaswamy

***Abstract*—In this paper, an efficient implementation of a 16 bit array hierarchy multiplier using full swing Gate Diffusion Input (GDI) logic is discussed. Hierarchy multiplier is attractive because of its ability to carry the multiplication operation within one clock cycle. The existing hierarchical multipliers occupy more area and suffer from accumulation delay of base multiplier output bits. These issues can be addressed by incorporating carry select adder based addition and the multiplier implementation using full swing GDI logic. The basic computation blocks involved in the multiplier are *AND* gate and carry propagate adder. They are implemented with using full swing GDI logic. Due to their reduced transistor count and less power consumption, this multiplier implementation leads to significant improvement compared with the existing implementations. The designed and existing array multipliers are simulated at 45 nm technology model and their power consumption and delay are calculated from the simulation results. It is validated that the proposed hierarchy array multiplier based on full swing GDI logic has 27% less energy consumption than the existing design. The results confirmed that implemented multiplier has shown better performance and can be used for signal and image processing.**

***Index Terms*— Full swing GDI logic, array multiplier, full adder, delay, digital circuit, hierarchy multiplier, carry select adder;**

Original Research Paper
DOI: 10.7251/ELS1721038M

I. INTRODUCTION

While the growth of electronics market has driven the Very Large Scale Integration (VLSI) industry towards very high integration density and system on chip, critical concerns have been arising to the severe increase in power consumption and area. High power consumption raises temperature profile of the chip and affects overall performance of the system. Moreover, the explosive growth in laptops and portable personal communication systems demand long battery life at the modest performance. This necessitates an intensive research in low power and low area IC design [1].

A Multiplier is a part of the processor that is widely used in digital devices such as computers, laptops, mobile phones and so forth. The various applications such as digital signal processing, image and video processing rely mainly in their multiplier performance. Also, multiplier is one of the major sources of power consumption in digital signal processor and microprocessor, etc. Therefore, the multipliers with high speed, lesser power consumption and low area are in great demand [2].

Hierarchical multipliers are considered as viable means for achieving orders of magnitude speed up in computer intensive applications through the use of fine grained parallelism. They are used in various fields of numerical and scientific computations, image processing, communication, cryptographic computation and so on [3-4].

In general, to design n bit hierarchical multiplier, four $n/2$ base multipliers are necessary which generate $2n$ bit output, where n represents hierarchical multiplier input width. It is noted that all the base multipliers are allowed to perform the task in parallel. Due to that, the performance of the hierarchy multiplier is determined from the accumulation delay of its base multipliers output bits. But this is a time consuming task as it requires more number of additions and considered a bottleneck for the hierarchy multiplier performance. In this work, an approach to perform this accumulation process is done by Carry Select Adder (CslA) to improve the performance. The following are the contributions discussed in the paper:

(i) For the simple hierarchy multiplier implementation, array multiplication scheme is chosen for base multiplier realization
(ii) To reduce the accumulation delay of base multiplier output bits, carry select adder is introduced
(iii) To realize the hierarchy multiplier with small area, it is implemented using full swing Gate Diffusion Input (GDI) logic

The rest of the paper is organized as follows: An overview of the GDI logic is described in Section 2. In Section 3, the explanations of the parallel adders are given whereas in Section 4 the architecture of the proposed hierarchy multiplier

Manuscript received 12 June 2016. Received in revised form 27 May 2017. Accepted for publication 28 June 2017.

Shoba Mohan is with the Department of electronics Engineering, Pondicherry University, India (Phone: +91-7598308967; e-mail: shobamalar@gmail.com).

Nakkeeran Rangaswamy is with the Department of electronics Engineering, Pondicherry University, India.

is described. The simulation results and discussion are given in Section 5 and finally, the Section 6 concludes the paper.

II. GDI LOGIC

The implementation of any digital circuits can be possible with Complementary Metal Oxide Semiconductor Logic (CMOS) [5]. It consists of both pull up (PMOS) and pull down (NMOS) transistor. Also, this logic needs an inverter to produce normal output from its complementary output. Though this logic has lesser power dissipation, it is suffer from more delay and area. To reduce the area i.e. (number of transistors required for the circuits implementation), Pass Transistor Logic (PTL) is invented [6]. But this logic has a drawback of threshold voltage problem which can be alleviated using a variant of pass transistor logic are complementary PTL called CPL, but the minimum requirement of generation of the complementary signal increases circuit overhead.

An alternative to CMOS logic, GDI logic is introduced, which is a low power design technique and offers the implementation of the various logic functions with fewer numbers of transistors [7]. The basic GDI cell is shown in Fig. 1. It looks like an inverter but it is not so. The source/drain both PMOS and NMOS transistor are tied at the diffusion input of P and N, respectively whereas in CMOS inverter, it is always tied at the V_{DD} and V_{SS}, respectively. The various logic functions realized using basic GDI cell is given in Table 1.

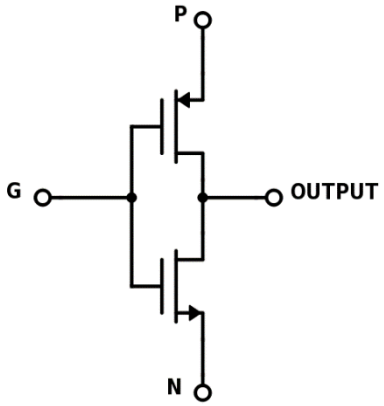

Fig. 1. Basic GDI cell [7]

Table 1. Various logic functions using GDI cell

TABLE 1
VARIOUS LOGIC FUNCTIONS USING GDI CELL

N	P	G	Output	Function
0	B	A	$\bar{A}$B	F1
B	1	A	$\bar{A}$+B	F2
1	B	A	A+B	OR
B	O	A	AB	AND
C	B	A	$\bar{A}$B+AC	MUX
0	1	A	$\bar{A}$	NOT

From the operational characteristics of GDI gates, it is concluded that they produce reduced output voltage for certain input combinations. This feature is beneficial for low power circuits. On the other hand, this may reduce noise margin and possible to increase the delay. Moreover, at low V_{DD} operation, the degraded output even may cause circuit malfunction. A simple technique for restoring output voltage level is the use of buffers. However, using buffers will cause propagation delay to increase proportionally to the number of cascaded stages [8]. Alternative to this, Ultra Low Power Diode (ULPD) is used, in which MOS transistor placed at the output terminal and can be realized as a diode [9]. An alternative to these techniques, full swing output voltage level can be retrieved by inserting proper transistor i.e. either NMOS/ PMOS depends on the voltage degradation. A set of full swing GDI gates and their operational characteristics are explained in [10]. As an example, the operation of XOR gate is explained.

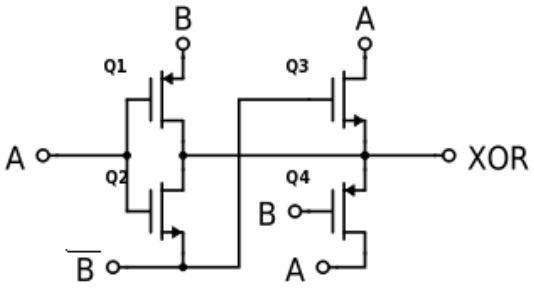

Fig. 2. Full swing GDI XOR gate [10]

The transistor level diagram of the XOR gate using full swing GDI logic is shown in Fig 2. The working mechanism of this gate is described here:

- Logic '*0*':

When *AB = 00*, Q1 and Q3 will be switched ON and other two transistors namely, Q4 and Q2 will be switched OFF. The output node is connected to *GND* potential through Q3 transistor. On the other hand, for the input combination of *AB = 11*, N1 transistor becomes switched ON and the remaining transistor are switched OFF. The output node is tied to *GND* potential.

- Logic *'1'*:

When *AB = 01*, the transistors Q1 and Q4 will be switched ON whereas Q2 and Q3 will be switched OFF state. It is well known that PMOS transistor is good at delivering strong '1' potential (V_{DD}). Likewise, for another input combination *AB =10*, the transistor Q4 and Q2 will be switched ON and the delivering of V_{DD} potential is taken care by the PMOS transistor P2.

From this discussion, it is understood that these components exhibit better performance in terms of delay, power consumption and area. Therefore, they can be chosen while implementing the array multiplier to improve performance.

III. PARALLEL ADDERS

Parallel adders are developed to minimize the delay involved in the binary addition task and are well suited for Very Large Scale Integration (VLSI) implementation. The

performance of these adders can be greatly influenced by the performance of their basic modules. The considered parallel adders are, ripple carry, carry select and carry look ahead adders. The basic modules of these parallel adders are Full Adder (FA) (for Ripple Carry Adder (RCA)), XOR and *AND* gate (for Carry Look Ahead (CLA)), FA and MUX (for Carry Select Adder (CslA)). Therefore, these basic modules are realized using Full Swing Gate Diffusion Input (full swing GDI) logic, discussed by the authors in the earlier work [10]. For detailed understanding of their working mechanisms, readers are directed to refer [10]. The transistor level representation of FA is given in Fig. 3.

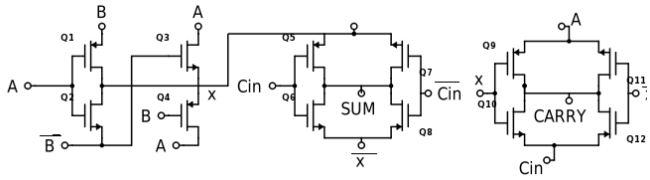

Fig. 3. Transistor level diagram of FA

A. Ripple Carry Adder

The RCA is *O (n)* time and *O (n)* area adders, where, *n* is the width of the operands. In the worst case, a carry can propagate from least significant bit position to the most significant bit position. Moreover, one stage of the RCA, the single full adder, determines the performance of RCA. Therefore, the delay of RCA can be decreased by implementing fast full adder. To achieve this, full adder based on full swing GDI logic is chosen as in [10]. Further, the carry propagation delay is reduced by minimizing carry propagation path or by performing pre-computation of carries.

B. Carry Look Ahead Adder

CLAs have become popular due to their high speed and modularity. They are *O (log n)* time and *O (n log n)* area adders. Consider the *n*-bit addition of two *n*- bit numbers $A= a_n, a_{n-1}, .., a_0$ and $B = b_n, b_{n-1},..., b_0$ resulting in the output sum $S = S_n, S_{n-1},.., S_0$ and carry out C_{out}.

The first stage in CLA computes the bit *generate* (G_i) and p*ropagate* (P_i) as follows

$$G_i=A_i.B_i \quad (1)$$

$$P_i = A_i \oplus B_i \quad (2)$$

These are then utilized to compute the final *sum* (S_i) and *carry* (C_{i+1}) bits,

$$S_i = P_i \oplus C_i \quad (3)$$

$$C_{i+1} = G_i \oplus P_i C_i \quad (4)$$

where $0 \le i \le n$.

The overall delay of carry look ahead adders is dominated by the delay of passing the carry in look ahead stages. The building blocks of CLA are *XOR* and *AND* gates. Moreover, the CLA performance is determined by the performance of them. Thus, the performance improvement in CLA is achieved by implementing those using full swing GDI logic in this paper.

C. Carry Select Adder

To minimize the delay due to carry propagation involved in RCA, CslA is evolved, in which, two additions are performed in parallel, one assuming C_{in} as *0* and other one as *1*. When the carry is known, finally the correct sum is selected. The pictorial representation of CslA is shown in Fig. 4(a). They are *O* (*2n*) area and *O* ($\sqrt{n}$) time adders. CslA is used in many computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a final carry to generate the sum. However, CslA is not area efficient because it uses multiple pairs of RCA to generate intermediate sum and carry for C_{in}= *0* and C_{in}=*1*.

The different techniques for minimizing the use of dual RCA in CslA is attempted in [13]-[15]. An interesting approach discussed in [13] is use of Binary to Excess 1 Converter (BEC) instead of RCA for C_{in}=*1* and its architecture is shown in Fig. 4(b). The inputs to BEC are as same as RCA, which is depicted in the Fig. 4b, as per ref. [13]. The BEC based CslA involves less logic resources than the conventional CslA. Further, the area reduction is possible in CslA with the help of sharing common Boolean logic expression for C_{in} *0* and *1* [14]. Though it requires less logic resources than the BEC based CslA, the carry propagation delay is larger.

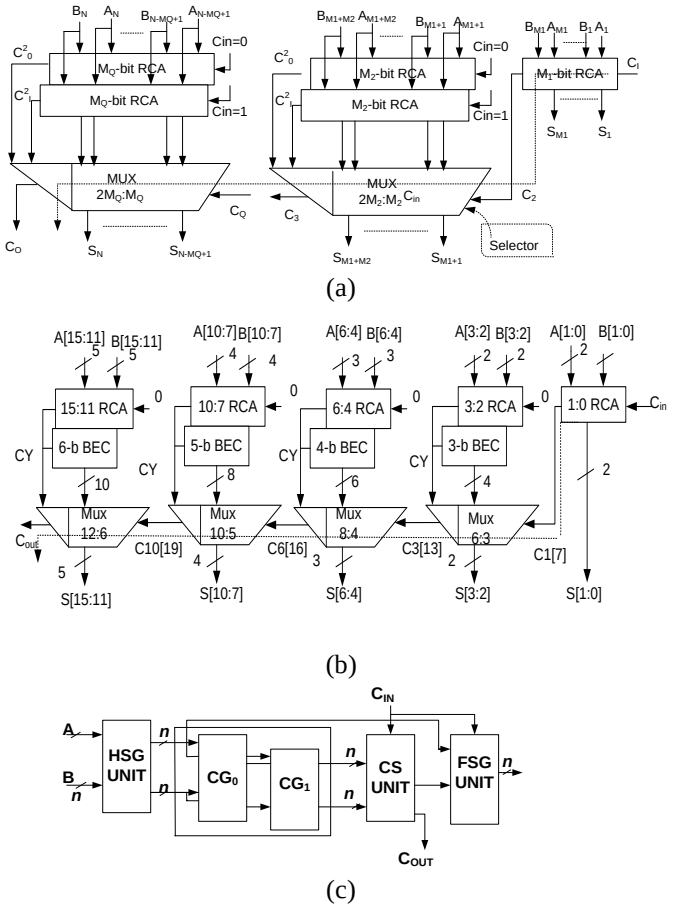

Fig. 4. CslA Adders (a) N-bit Conventional CslA in [11] (b) 16 bit BEC-CslA in [13] and (c) N- bit CslA in [15]

CslA design is simplified based on logic reformulation and optimization of carry generator module [15]. This design possesses less area and delay than the conventional CslA design. This adder schematic is given in Fig. 4(c). The performance of CslA design can be improved by proper implementation of their basic modules such as MUX and FA. Therefore, these are designed using full swing GDI logic. In this paper, the conventional CslA in [11], BEC CslA discussed in [13] and CslA given in [15] are implemented and their performance improvements are studied through simulation.

IV. Hierarchy Multiplier

Multipliers with large width are required for the implementation of cryptography and error correction circuits for more reliable transmission over highly insecure and/or noisy channels in networking and multimedia applications. The hierarchical principle helps to realize fast large bit multiplier, except that it requires a large width adder for performing the addition task, which poses limitation on the performance and increases area of the designed multiplier [16-18].

Over the last few decades, a lot of works have been dedicated, at the algorithmic and implementation level, to improve the performance of hierarchical multiplier. The conventional hierarchy multiplier architecture is shown in Fig. 5(a). The multiplier inputs are *X*, *Y* of *n* bit width and produces the output *P* of *2n* bit. First, the inputs *X* and *Y* are divided into equal two halves namely, X_H and X_L, Y_H and Y_L and they are multiplied by base multiplier. As seen in Fig. 5 (a), the base multipliers are used for the multiplication of (X_L and Y_L), (X_H and Y_L), (X_L and Y_H) and (X_H and Y_H). Once these multiplication processes are over, their output bits will form a carry save array which is processed by Carry Save Adder (CSA) resulting with two rows of *16* bit output. These bits are further added with the help of Carry Propagate Adder (CPA) to produce the multiplier output bits.

The delay in the addition process of the hierarchy multiplier is reduced with the parallel execution of ripple carry adder [16]. However, this method requires twice the number of adders thus results in increased area. On the other hand, the delay is reduced with the deployment of carry look ahead adder for the addition process but this increases the interconnection complexity [17]. Not only delay and area, the power consumption of the hierarchy multiplier also has to be reduced because the existing designs appending more zeros to equalize the number of bits in order to make them suitable for parallel computation [18]. This might increase the spurious activities and thus increases the power consumption. The above mentioned issues in the existing hierarchy multiplier can be addressed by

(i) Performing the final addition using proposed carry select adder,

(ii) Implementing the proposed hierarchy multiplier using full swing GDI logic as carried out in this paper.

The architecture of the proposed hierarchy multiplier is given in Fig. 5(b). The carry propagate adder is replaced by carry select adder to improve the computation time.

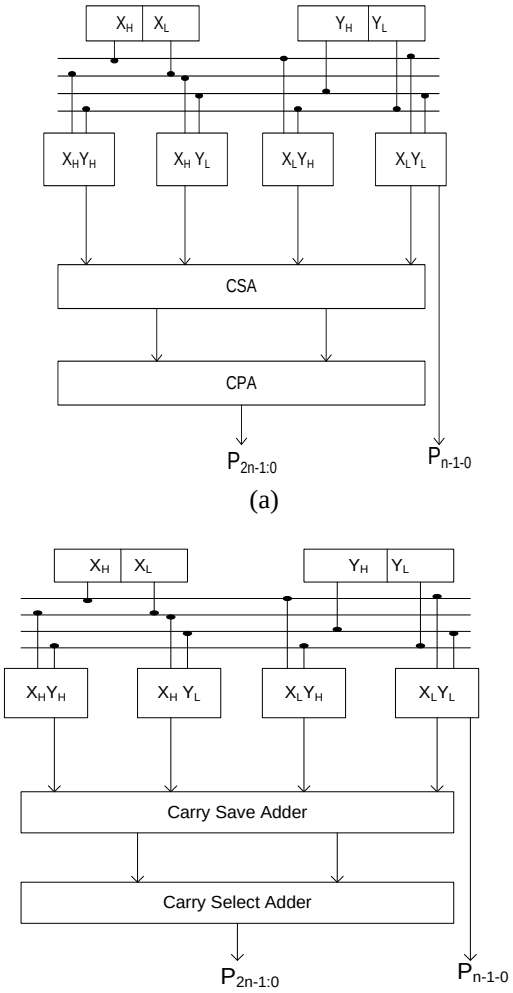

Fig. 5. Hierarchy Multiplier Architecture (a) Conventional and (b) Proposed

A. Base Multiplier

The array multiplier is chosen as base multiplier, which uses full adders for partial products reduction. Though it has a regular layout structure, the delay is increasing with increase in number of input bits. Since the delay of array multiplier is mainly determined from this adder delay, which might be decreased by incorporating hierarchy principle based multiplication. The architecture of a 4x4 array multiplier for partial products reduction is given in Fig. 6. In the architecture, *a* and *b* are multiplier input operands each having 4 bits. The partial products are generated from *AND* gates, acted as inputs to these adders. Once the partial products accumulation is completed, the array multiplier output bits $P_7,\ldots, P_0$ become available. It is evident from the multiplier architecture, its power consumption and the area is determined by the number of transistor of all adders in the array. Since the basic components of the array multiplier are *AND* gate and full adder, this performance can be improved by utilizing the better circuits for them. Generally, the partial products of multiplication are generated by *AND* gates. For *n* bit

multiplier, it requires n^2 *AND* gates. Because the *AND* gate based on CMOS and GDI uses 6 transistors, it needs $6n^2$ transistors, to construct the *AND* gate array in the *n* bit array multiplier. But the full swing GDI based *AND* gate uses 5 transistors only. Thus, results in decreased number of transistors. Another component of multiplier i.e., full adder, required for *n* bit multiplier is $(n\text{-}1)n$, which is realized in CMOS, GDI and full swing GDI uses 28, 24 and 18 number of transistors, respectively. As a result of these modifications in the multiplier implementation, its performance is improved. The reduced number of devices not only offers the reduction in delay, the unnecessary switching activities are also minimized, thereby minimizing the power consumption as well.

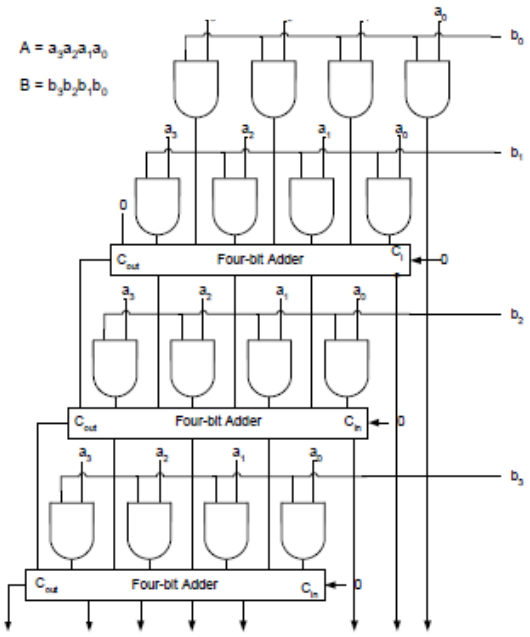

Fig. 6. 4x4 Array multiplier

V. Simulation Results and Discussion

A. Parallel Adder

In this Section, the simulation results of the parallel adders based on CMOS, GDI and full swing GDI logic are presented and the performance of them is compared. During the evaluation of these adders, the performance metrics such as area, delay, power consumption and Power Delay Product (PDP) are taken into account. The simulations are performed at 45 nm freePDK technology with a supply voltage (V_{DD}) of 1.1 V using Cadence Virtuoso tool. Typical transistor sizes, i.e., $(W/L)_p$=120 nm/45 nm and $(W/L)_n$=240 nm/45 nm are used [19]. After the completion of simulation of parallel adders, the layout is generated for each of them and subjected to Design Rule Check (DRC) then Layout Versus Schematic (LVS) check before the extraction of parasitic. Subsequently, the extracted parasitic file is back annotated to perform the post layout simulation. In the simulation environment, each input is driven by buffered signal and each output is loaded with buffer. Power and delay of the buffers are included in the power and delay calculations of the whole circuit.

Delay: The delay is measured by accounting the time from the 50% of the input voltage swing to 50% of the output voltage swing for each transition. The maximum delay is treated as worst case delay. The delay computed through simulation, for all the adder structures are plotted in Fig. 7(a). As it is expected, CLA structures have smaller delay compared to those other four adders due to the parallel computation of their carries. On the other hand, RCA has the highest delay due to its serial structure. However, RCA implemented based on full swing GDI adder, as reported in [10] has shown 12% and 6% speed improvement than CMOS and GDI adders, respectively.

The critical path delay of CslA is smaller than that of RCA due to the skipping of carry propagation. Further, the percentage of delay reduction in conventional CslA, BEC CslA in [13] and CslA in [15], which are implemented based on full swing GDI is 15, 27 and 20 than CMOS based implementation of those adders. The implementation of basic modules of CslA such as XOR, MUX and FA in full swing GDI logic can able to provide significant speed improvement than attained in other parallel adders like RCA and CLA.

Power Consumption: The power consumption of any circuit mainly depends on the switching activities of node and wire capacitances. The power consumed by the parallel adders are computed through simulation and also presented in Fig. 7(b). The results indicate that the CLA and CslA have more power consumption than that of RCA. The minimum power consumption is witnessed in RCA owing to its simple and regular structure while CLA consumes more power due to its dense wiring tracks. However, the power consumption of the CLA based on full swing GDI reduced by 30% comparing to CMOS based design. As shown in Fig. 7(b), the power consumption of adders based on full swing GDI gates is decreased almost same for all the adders and the results reveal, on average, 35% improvement is achieved over CMOS logic. Comparing with conventional GDI, full swing GDI minimizes the power consumption in parallel adders, on average, 30% by maintaining full swing voltage at intermediate nodes, which reduces the spurious transitions of the adder.

Area: The layout is drawn for all these implemented adders. The area is evaluated from their layout and it is plotted in Fig. 7(c). From the obtained results, it is witnessed that full swing GDI based RCA has less area whereas more area belongs to CMOS based CLA adder. Since the single FA realized by full swing GDI has less area than either CMOS or GDI logic, which might be a reason that overall area of RCA becomes lesser. Likewise, in the CslA designs, 26% area savings is achieved in full swing GDI based designs than those are implemented in CMOS logic. Similarly, the percentage of area reduction using full swing GDI based CLA adder is 17 and 13, respectively more than CMOS and GDI logic.

PDP: The power delay product of the parallel adders for CMOS, GDI and full swing GDI logic are plotted in Fig. 7(d). Among the adders discussed, the worst and the best PDP belongs to full swing GDI based CslA in [15] and conventional CslA based on CMOS, respectively.

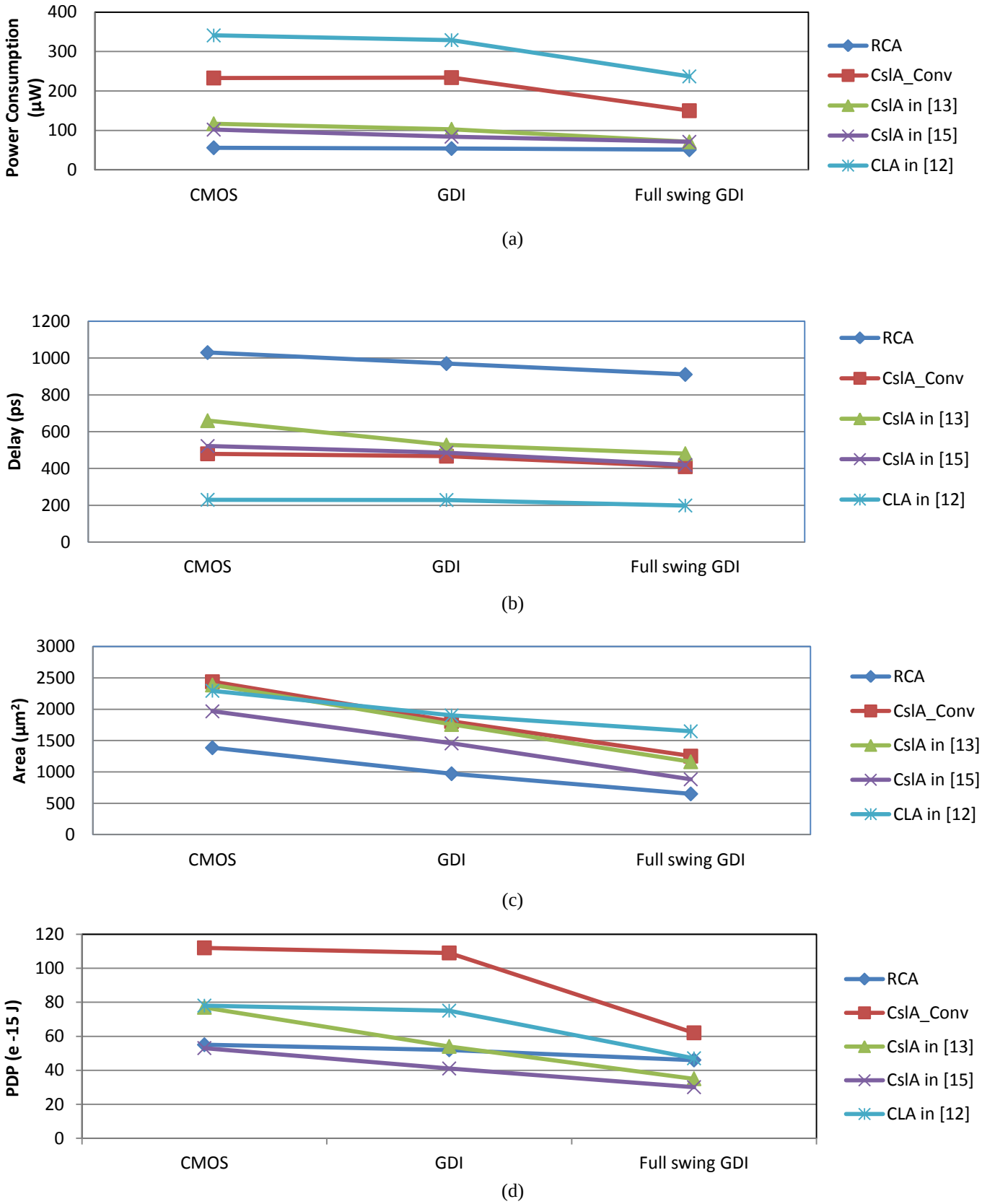

Fig. 7 Simulation results of parallel adders (a) Delay (b) Power Consumption (c) Area and (d) PDP

However, the PDP of conventional CslA is reduced in full swing GDI by 45% and 43% than CMOS and GDI, respectively. Similarly, in the CLA and RCA operated with lesser PDP in full swing GDI by 40% and 16%, respectively than CMOS. Also, it is examined from the obtained results of PDP of parallel adders, CslA implemented with full swing GDI logic has small PDP with acceptable speed and hence, they can be a proper choice while designing high performance and low power applications.

Sensitive to Process Variation: In order to evaluate the sensitivity of the designs to local and global process variations Monte Carlo simulations have been carried out for parallel adders. The simulations have carried out for 1000 runs. The variations in power consumption, delay and PDP with respect to the process variations are depicted in Fig. 8. As expected, the full swing GDI based parallel adders have better immunity to process variation compared with others. The parallel adders, based on CMOS have variation in delay, power consumption and PDP as, on average, 3%, where as in full swing GDI based, the variations are approximately about 1.

*Adder Efficiency: A*n approach of comparing different adders by defining Merit Factor (MF) based on the performance and the reliability parameters is discussed in [1]. The speed and energy consumption are the key parameters for determining the efficiency of the adders design, so the product of delay and power are considered as the merit factor. The expression merit

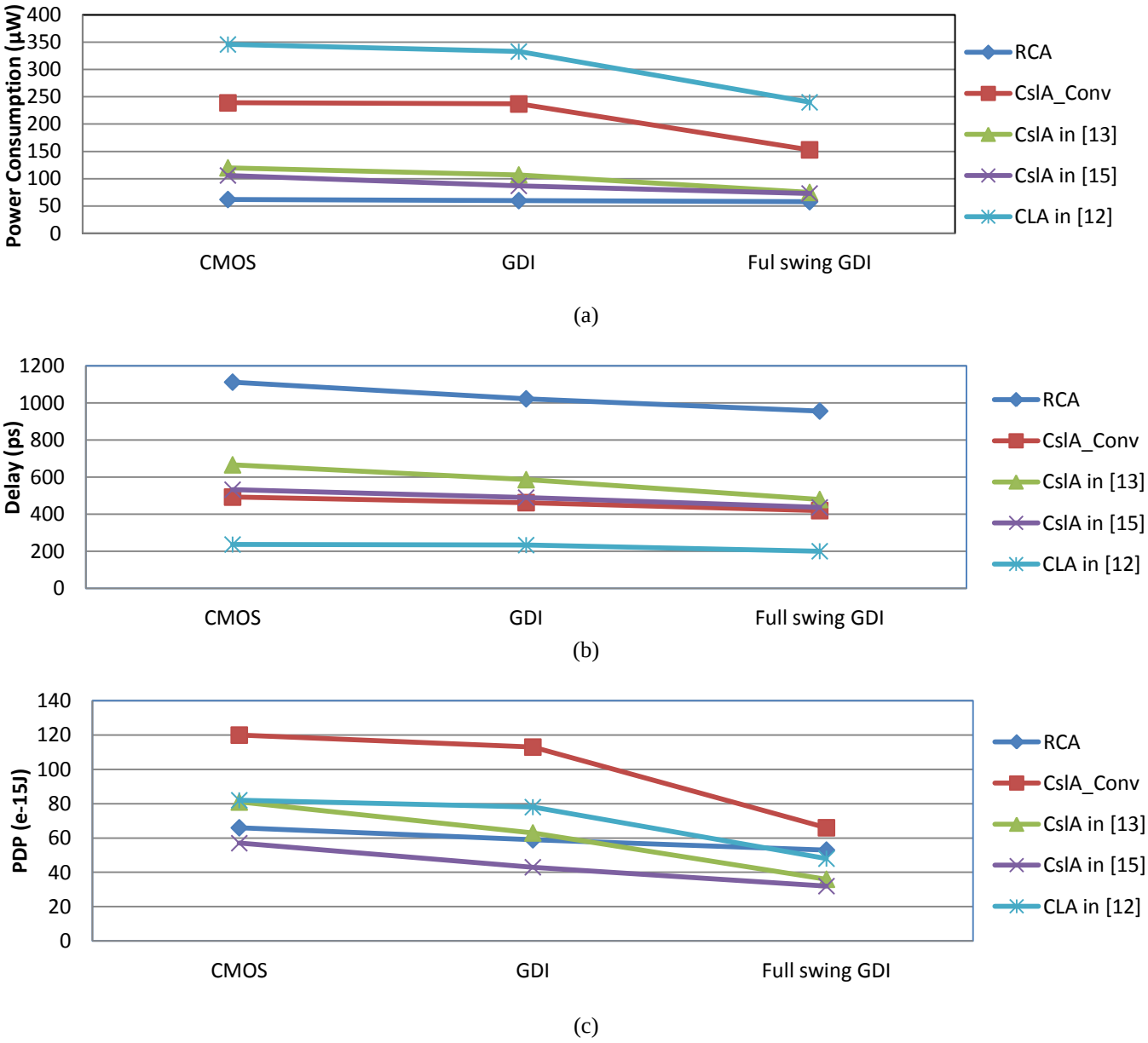

Fig. 8 Monte Carlo Simulation Results of Parallel Adders (a) Power Consumption (b) Delay and (c) PDP

factor is given in Eq. (5).

$$MF = \frac{1}{D_{norm} \cdot P_{norm} \cdot \Delta D_{norm,pv} \cdot \Delta P_{norm,pv}} \quad (5)$$

The delay and power consumption of the adders may be normalised by dividing them to the maximum value of power consumption and delay of particular adder among all the logic styles. The normalised delay and power consumption, represented as D_{norm} and $P_{norm,}$ respectively. In addition to that, normalized delay and power consumption changes (variations) due to process variations, represented as ($\Delta D_{norm,\ pv}$) and ($\Delta P_{norm,\ pv}$), respectively, are included in MF to evaluate the reliability of the designed adders. The amount of ($\Delta D_{norm,\ pv}$) for each logic style is calculated by dividing the delay variations due to process variation with the maximum value of delay variation among all the logic styles for a particular adder. The same procedure is applied for ($\Delta P_{norm,\ pv}$) computation. For merit function defined above, higher MF corresponds to better adder design.

The merit factor for all the adders versus different logic styles namely, CMOS, GDI and full swing GDI is shown in Fig. 9. It is observed that, the low and high value of MF corresponds to full swing GDI based CslA in [15] adder and CMOS based CLA adder. Moreover, full swing GDI logic based parallel adders namely, CslA in [15] and RCA has shown MF improvement of 80% and 36%, respectively than the realization in CMOS logic. From the obtained results, it is concluded that the adder efficiency can be improved significantly with the help of full swing GDI logic implementation.

The performance improvement of parallel adder structures such as RCA, CslA and CLA with the help of full swing GDI logic is attempted. The basic modules of these adder such as XOR, AND, MUX and full adder are realised using full swing GDI logic. It is observed that, the full swing GDI based RCA, CslA and CLA have shown speed improvement in terms of 12 %, 27 % and 14%, respectively than CMOS logic. Likewise, the amount of area reduction achieved in RCA, CslA and CLA is 53%, 28% and 17%, respectively than CMOS logic. Among the parallel adders, the CslA adders based on full swing GDI logic have 45% PDP improvement than their existing implementation.

Finally, the adder efficiency is measured for all the simulated adders based on the merit factor, which depends on delay and power delay product. From the efficiency results, it is concluded that the full swing GDI based parallel adders namely; CslA adder discussed in [15] has shown merit factor improvement by 80% compared to that CMOS based implementation of the same adder. From the discussion of the performance improvement in parallel adders based on full swing GDI logic, CslA adders have 45% improvement in PDP than that of RCA and CLA adders. Therefore, they can be used in the place of adders while implementing multipliers for improving their performance.

B. Hierarchy Multiplier

In this Section, the simulation results of the conventional and the proposed hierarchy array multiplier are presented.

Delay: The delay is measured from the 50% of the input voltage swing to 50% of the output voltage swing for each transition. The maximum delay is treated as worst case delay. The delay results of the simulated multipliers are given in Table 2. The hierarchy multiplier based on full swing GDI gates has 7% lesser delay than conventional design. This is attained due to the deployment of CslA in the hierarchy multiplier architecture thus reduces the base multiplier accumulation delay.

Power Consumption: The power consumed by the multipliers are computed through simulation and also presented in Table 2. From the results, it is observed that the multiplier based on full swing GDI logic consumes less power than the conventional solution discussed in [3]. This is achieved due to the elimination of spurious transitions involved in the multiplier. Also, the full swing output at intermediate nodes minimizes the unnecessary switching of the transistor which

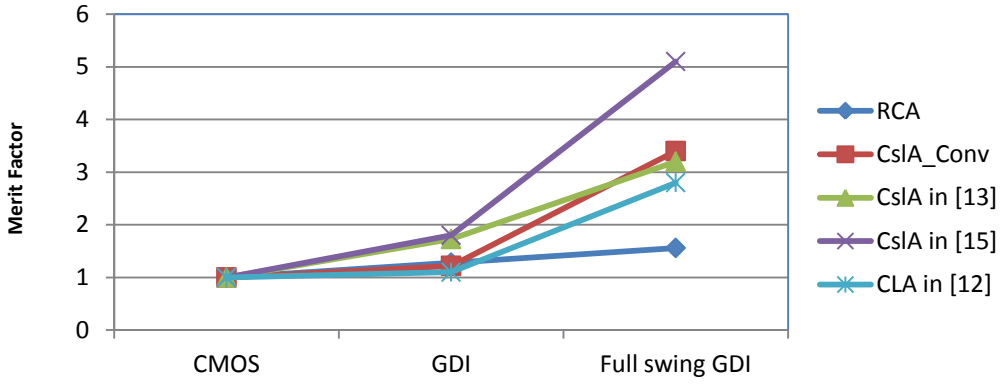

Fig. 9. Merit factor of parallel adder

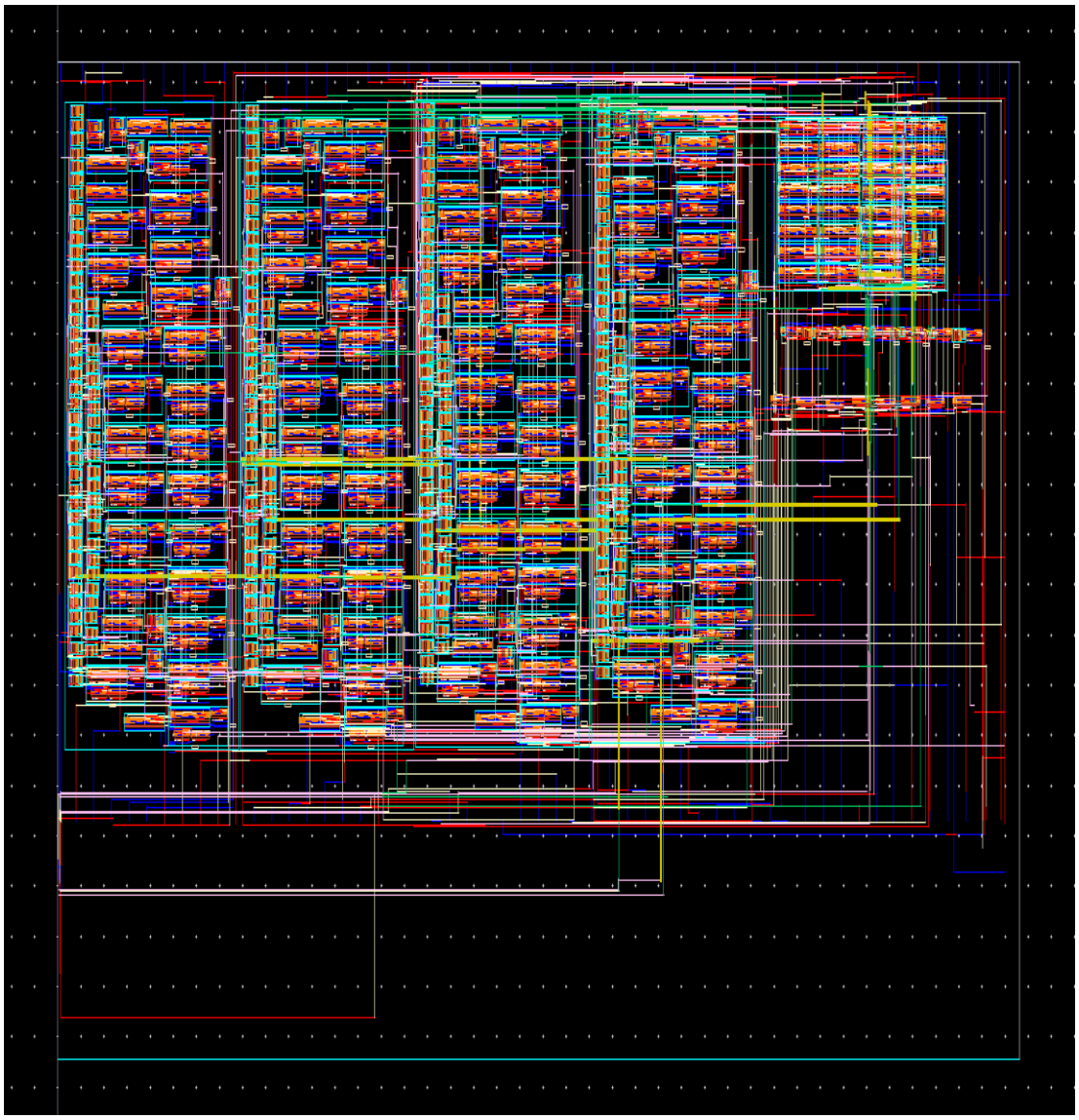

Fig. 10. Layout of the hierarchy multiplier architecture

would results in the power consumption minimization. The power saving is possible with full swing GDI based hierarchy multiplier design is 21% than existing multiplier.

TABLE 2
SIMULATION RESULTS OF 16 BIT ARRAY MULTIPLIER

Multiplier	Delay (ps)	Power Consumption (µW)	PDP (e-15 J)	Area (μm^2)
Conventional [3]	882	70	62	17857
Proposed	824	55	45	16681

Area: The layouts are drawn for all the simulated multiplier and the area is calculated from them and listed in Table 2. From the obtained results, it is observed that the proposed multiplier requires 7% less area compared with the existing hierarchy multiplier. This is attained due to the implementation using full swing GDI logic, which in turn minimises the transistor count thus results the small area. The layout of the proposed multiplier is given in Fig. 10.

*PDP***:** From the simulated results, it is observed that, proposed multiplier has smaller PDP or energy consumption. It is observed from the results that conventional design operates with more energy consumption than the proposed one.

Sensitive to Process Variation: A study of circuits performance under the local and global process variations is carried through Monte Carlo simulations with thousand runs (*N*=1000) and observed that the proposed multiplier is able to sustain the same performance with or without process variations.

VI. CONCLUSION

The existing implementation of array multiplier based hierarchy multiplication lacks in terms of area and delay, which is due to the requirement of more transistor count for its base components such as *AND* gate and adder. To overcome these drawbacks, hierarchy multiplier is designed with CslA

which is further implemented using full swing GDI logic. An investigation of various parallel adder architectures performance is carried out and it is concluded that CslA possess better performance than conventional one. The introduction of CslA in hierarchy multiplier reduced its delay significantly. The performance of the multipliers is analyzed using SPICE simulation at 45 nm technology models. The performance parameters like delay and power consumption of the multipliers are measured from simulation results. From the simulation results, it is understood that the implemented multiplier design gives smaller power delay product and number of transistor comparing to the design found in the literature.

ACKNOWLEDGMENT

The authors would like to thank the VIT University, Vellore, India for providing support to carry out some of the simulation works at Integrated Circuit Design Laboratory.

REFERENCES

[1] M. Bahadori, M. Kamal, A. Afsail-kusha and M. Pedram, A Comparative study on performance and reliablity of 32 bit binary adders, *Integration, the VLSI Journal*, Vol. 53, No. 1, 2016, pp. 54-67.

[2] M. Valinataj, Fault-tolerant carry look ahead adder architectures robust to multiplie simultaneous errors, *Microelctronics Reliability*, Vol. 55, No. 12, 2015, pp. 2845-2857.

[3] G. Quan, J. P. Davis, S. Devarkal and D.A. Buell, High level synthesis for large bit width multipliers on FPGAs: A case study. *In proceedings of the international conference on hardware/software codesign and system synthesis*, pp. 213-218, 2005.

[4] Z. Zakaria and S. A. Abbasi, Optimized multiplier based upon 6 input LUTs and Vedic mathematics, *World Academic of Science, Engineering and Technology*, Vol. 7, 2013, pp. 26-30.

[5] S. Goel, A. Kumar and M.A. Bayoumi, Design of robust,energy-efficient full adders for deep-submicrometer design using hybrid-CMOS logic style, *IEEE Trans. on VLSI Syst*., Vol. 14 No.12, 2006, pp. 82-94.

[6] S. Purohit and M. Margala, Investigating the impact of logic and circuit implementation for full adder performance, *IEEE Trans. on VLSI Syst*, Vol. 20, No. 7, 2012, pp. 1327-1331.

[7] A. Morgenshtein, A. Fish and I. A. Wagner, Gate-Diffusion Input (GDI) – A power-efficient method for digital combinatorial circuits, *IEEE Trans. on VLSI Syst*., Vol. 10, No. 5, 2002, pp. 566-581.

[8] A. Morgenshtein, I. Shwartz and A. Fish, Gate Diffusion Input (GDI) Logic in standard CMOS nanoscale process, Proc. of *IEEE Convention of Electrical and Electronics Engineers* in Israel, pp.776-780, 2010.

[9] V. Foroutan, M. Teheri, K. Navi and A. Mazreah, Design of two low power full adder using GDI structure and hybrid CMOS logic style, *Integration, the VLSI Journal*, Vol. 47, No.1, 2014, pp. 48-61.

[10] M. Shoba and R. Nakkeeran, GDI based full adders for energy efficient arithmetic applications, *Engineering Science and Technology, an International Journal*, Vol. 19, No.1, 2016, pp. 485-496.

[11] N. H. E. Weste and D. Harris, CMOS VLSI Design, 2nd ed, Pearson Education, 2005.

[12] A. Morgenshtein, I. Shwartz and A. Fish, Full swing Gate Diffusion Input (GDI) logic –case study for low power CLA adder design, *Integration, the VLSI Journal*, Vol. 47, No.1, 2014, pp. 62-70.

[13] B. Ramkumar and H. M. Kittur, Low-power and area-efficient carry select adder, *IEEE Trans.VLSI. Syst*., Vol. 20, No. 2, 2012, pp. 371-375.

[14] S. Y. Sun, Z. X. Zhang, and X. J. Jin, High-performance carry select adder using fast all-one finding logic, *Second Asia International Conference on Modelling Simulatio*, pp. 1–5, 2008.

[15] B. K. Mohanty and S. K. Patel, Area-delay-power efficient carry-select adder, *IEEE Trans. Circuits Syst. I Regul. Pap*., Vol. 61, No.6, 2014, pp. 418-422.

[16] J. Shi, G. Jing, Z. Di and S. Yang, The design and implementation of reconfigurable multiplier with high flexibility, *In proceedings of the international conference on electronics, communications and control*, pp. 1095-1098, 2011.

[17] S. A. Abbasi, A. R. M. Zulhelmi and Alamoud, FPGA design, simulation and protyping of 32 bit pipeline multiplier based on Vedic mathematics, *IEICE Electronics Express*, Vol. *12*, 2015, pp. 1-12.

[18] S. Quan, Q. Qiang, C. L. Wey, (2005). A novel reconfigurable architecture of low power unsigned multiplier for digital signal processing, *In proceedings of the international symposium on circuits and systems*, pp.3327-3330, 2005.

[19] https://www.eda.ncsu.edu/wiki/FreePDK45