JOURNAL OF NANO- AND ELECTRONIC PHYSICS
Vol. **10** No 2, 02022(8pp) (2018)

ЖУРНАЛ НАНО- ТА ЕЛЕКТРОННОЇ ФІЗИКИ
Том **10** № 2, 02022(8cc) (2018)

Energy Efficient Design of Four-operand Multiplier Architecture using CNTFET Technology

N. Charmchi, M.R. Reshadinezhad*

Faculty of Computer Engineering, University of Isfahan, 8174673441 Isfahan, Iran

(Received 21 August 2017; revised manuscript received 28 April 2018; published online 29 April 2018)

Multiplication is an essential part of digital arithmetic, due to its application in video and voice processing, FIR filters, cryptography and other related concepts. Reducing the power consumption and increasing the speed of multipliers will affect the performance of any VLSI system. An approach to accomplish the desired objective for the researchers is applying nano-technologies in implementing VLSI circuits. Carbon nanotube technology is an appropriate option among emerging nano-devices, due to its similarities to the preceding technology, MOSFET. Three new architectures are proposed for a four-bit four-operand multiplier. These multipliers and the conventional four-bit four-operand multiplier are designed, implemented and simulated through carbon nanotube field effect transistors. Evaluations and comparisons are run through HSPICE simulator, through using carbon nanotube technology. These multipliers outperform the common four-operand multiplication run on computers nowadays, referred to as conventional multiplier in this article.

Keywords: Multi-operand multipliers, Fast multipliers, CNTFET nanotechnology multipliers, CNTFET full adders, Parallel multipliers.

DOI: 10.21272/jnep.10(2).02022 PACS numbers: 84.30. – r, 85.35.Kt

1. INTRODUCTION

Multiplication is one of the fundamental operations in processors. The time needed to perform multiplication increases as the length of the operands increase, which leads to a rise in circuit complexity. There exist several studies in the field of multipliers [1, 2]. These researches reveal that multipliers could be implemented through three distinct stages. The first stage includes AND gates for partial product generation, where, some designers apply modified booth algorithm in order to cut partial products to half [1, 4]. In the second stage, partial products are reduced to two vectors of *sum* and *carry* using partial product reduction methods such as Wallace tree, Dadda tree and compressors [1, 2, 5]. The last stage consists of a carry propagating adder (CPA) which adds the two vectors generated in the second stage; hence, the final multiplication product [5, 6].

In computers in use, a four-bit four-operand multiplication is computed through three two-operand multipliers' set in series. The three-operand multiplication is presented by [7] at gate level. A three-operand multiplication at transistor level is proposed by [8].

In different processor architectures, multipliers and adders are considered as the main components of the processors, in a sense that any improvement in their function would lead to an increase in program speed. Hence, applying an energy efficient multi-operand multiplier is essential. Here is an example of using four-operand multiplier in an exponential operation. In arithmetic, there is a common method for computing g^e, that is multiplying g to itself for e times. Assume that $a = g^{15}$. This can be written as:

$$\begin{aligned} &g \to g^2 \to g^3 \to \ldots \to g^{15} \\ &g^{15} = g^4 \times g^4 \times g^4 \times g^3 \end{aligned} \qquad (1\text{-}1)$$

In order to obtain the final multiplication result, 14 multiplication operations are required. The result could be achieved by applying four-operand multipliers using Eq. (1-1). In the first step, three four-operand multipliers and a three-operand multiplier is required. Then, the final result is calculated through a four-operand multiplier. In exponentiation operations with higher powers used as a key in cryptography, it is evident that the higher-operand multipliers could be applied in order to perform the operation.

Multiplication is one of the most time-consuming operations in arithmetic units. Researchers suggest some techniques for reducing the chip size, delay and power consumption of the circuits. In this regard, one of the approaches is attained through applying carbon nanotube technology in implementing arithmetic circuits.

According to the Moore's law, the number of transistors doubles every 18-24 months intervals. This phenomenon is named scaling and results in reducing the chip size; consequently, diminishing the channel length. Therefore, it leads to some challenges in MOSFET technology such as the quantum effects, high lithography costs, ICs eating, increased leakage currents, on-current increase difficulty, large parameter variations, low reliability and yield as well as an increase in manufacturing cost [9]. New emerging technologies are introduced, in order to overcome these challenges and improve circuits' performance. Quantum-dot Cellular Automata (QCA), Single Electron Transistor (SET) and Carbon Nanotube Field Effect Transistor (CNTFET) are examples of nano-technologies [10]. CNTFET is the leading technology among nano-devices in comparison to MOSFET technology, due to its similar electrical characteristics. The implementation of the first nano-computer in Stanford University can be regarded as a breakthrough in replacing of CMOS through CNTFET. The carbon nanotube circuits provide a big chance for the researchers with respect to achieving low power consumption and

* m.reshadinezhad@eng.ui.ac.ir

high speed in comparison with the circuits made of silicone. Lombardi et al. compared MOSFET and CNTFET technologies and revealed that applying carbon nanotube technology reduces power consumption and delay in arithmetic circuits [9]. According to the studies, CNTFET-based circuits outperform MOSFET-based counterparts and implementing a four-operand multiplier through carbon nanotube technology obtains better results than a MOSFET-based four-operand multiplier.

In this article, three new architectures for a four-bit four-operand multiplication are proposed. The proposed designs are compared with the four-bit four-operand multiplication performed on conventional processors. As mentioned, multiplication is based on adding partial products; therefore, applying an efficient full adder enhances the multipliers' performance. To materialize this objective, a full adder cell with an optimized performance must be proposed with respect to multipliers architectures.

This article is organized in six sections: in Section 2 a review of Carbon Nanotube Field Effect Transistors is introduced in brief. The previous multipliers are presented in Section 3. The conventional CNTFET-based multipliers are illustrated in Section 4. The three new architectures proposed for a four-bit four-operand multiplier are presented in Section 5. The simulation results in various situations and their comparison with the conventional state-of-the-art multiplier is presented in Section 6 and the article is concluded in Section 7.

2. REVIEW OF CARBON NANOTUBE AND CARBON NANOTUBE FIELD EFFECT TRANSISTORS

Carbon nanotubes (CNTs) have become an intriguing issue among researchers, in a way that these nano-devices have turned into an alternative technology for MOSFETs. This CNT was first discovered by Ijima in 1991 [11]. CNT is a sheet of graphene (an allotrope of graphite) rolled up and shaped as a tube. According to the number of the tubes, the CNTs are categorized in two groups of Single Walled Carbon Nanotube (SWCNT) and Multi Walled Carbon Nanotube (MWCNT). The MWCNTs are composed of 2 to 30 nested concentric graphene layers (Fig. 1), while SWCNTs consist of one graphene layer [12].

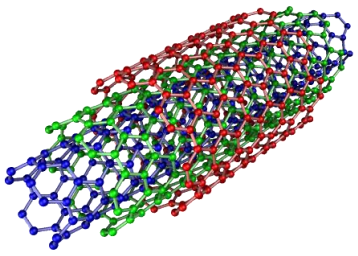


Fig. 1 – Multi walled Carbon Nanotube (MWCNT)

Every CNT has a specific vector defined as $Ch = n\bar{a}_1 + m\bar{a}_2$, named the Chirality vector. The (*n, m*) are Chirality numbers and $\left[\bar{a}_1, \bar{a}_2\right]$ are the unit vectors. Depending on *n* and *m* the SWCNTs can be categorized in three types: $m = 0$ or $n = 0$ which makes the CNT zigzag, $n = m$ which makes the CNT armchair and any other value for *n* and *m* makes it chiral (Fig. 2). According to the direction of the Chirality vector, the graphene is rolled and the nanotube is fabricated [13].

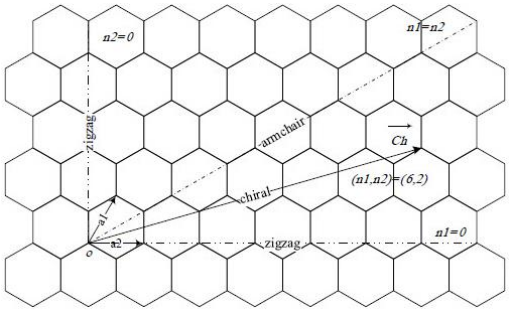


Fig. 2 – Sheet of graphene to be formed as CNT

The CNT diameter – measured in *nm* is calculated through Eq. (1).

$$D_{CNT} = \frac{a}{\pi}\sqrt{n^2 + nm + m^2} = 0.0783\sqrt{n^2 + nm + m^2} \quad (2\text{-}1)$$

where, parameter $a = \sqrt{3}a_0$ is the carbon to carbon distance and $a_0 = 0.142$ nm is the length of the carbon-carbon bond. The threshold voltage of a CNTFET is determined through adjusting the nanotube diameter; therefore, it is possible to turn on the CNTFET at a required voltage and design the appropriate circuit with better performance. The threshold voltage is computed through Eq. (2).

$$V_{th} = \frac{E_g}{2e} = \frac{\sqrt{3}}{3}\frac{a\nu_\pi}{eD_{CNT}} \approx \frac{0.43}{D_{CNT}} \quad (2\text{-}2)$$

where, ν_π is the π-π bond energy in the tight bonding model, equal to 3.033 eV, *e* is the unit electron charge, D_{CNT} is the nanotube diameter and E_g is the band gap. As observed in Eq. (2-2), the threshold voltage of a CNTFET is approximated as the inverse function of the nanotube diameter. If *m* is assumed to be zero in Eq. (2-1), another equation will be developed for comparing the two CNTFETs.

$$\frac{V_{th1}}{V_{th2}} = \frac{D_{CNT2}}{D_{CNT1}} = \frac{n_2}{n_1}$$

Carbon nanotube transistors have one dimensional structure that causes ballistic transport in them. Furthermore, the PCNTFET and NCNTFET have the same mobility and a similar geometry; hence, a unified drive capability [14].

According to high speed, low power consumption and appropriate performance of the circuits, this technology is adopted to implement the three proposed architectures. Moreover, due to various advantages and disadvantages of available three types of CNTFETs (SB-CNTFET, T-CNTFET and MOSFET-like CNTFET) and considering their performances, the MOSFET-like CNTFET is a proper option applied in the proposed architectures in this article.

3. PREVIOUS WORKS

3.1 The Two-operand Multipliers

Several articles have introduced various techniques of multiplier implementations [1-3]. Itoh et al. presented a multiplier in rectangular-style Wallace tree. Their objective is to decrease the chip size; hence, a reduction in layout cost and chip fabrication [2].

A low power Wallace multiplier based on wide counters was proposed by Abed et al. in 2011. This multiplier operated similar to that of the Wallace multiplier with a difference in the partial product reduction stage. At this stage, this proposed multiplier applied counters, instead of full adders; therefore, this circuit has fewer gates, consequently, consumed less power. This method is superior in the case of carry propagation, because counters do not have input carry, while compressors have [15].

A four-bit two-operand CNTFET-based multiplier is introduced in 2015. Mhaske et al. applied a low complexity Wallace multiplier that was previously introduced in [16]. This method, due to a reduction in number of full adders utilized in the second step of the multiplication has lower power consumption compared with the typical Wallace [3].

Charmchi et al. introduced a high speed two-operand multiplier through CNTFETs. The presented design has lower delay and PDP in comparison with its preceding counterpart [17]. Reshadinezhad et al. introduced a four-operand multiplier presented in [18], which is designed at transistor level for which three methods are introduced and extended in this article.

3.2 The Multi-operand Multipliers

Three methods are suggested for a four-bit three-operand multiplier by [7]. All simulations are performed at gate level and the authors confirmed that their proposed designs have higher speed in comparison with other designs [7].

Implementation of a three-operand multiplier through CNT technology is introduced in [8]. The proposed design is compared with its conventional counterpart and the results indicate the superiority of the proposed multiplier [8].

4. THE FOUR-BIT FOUR-OPERAND CONVENTIONAL MULTIPLICATION

In computers in use, a processor multiplies the operands A (a_{n-1} … a_1a_0) and B (b_{n-1} … b_1b_0) which results in a $2n$ bit product. Next, the previous result is multiplied to operand C (c_{n-1} … c_1c_0) and a $3n$ bit is obtained. Eventually, this $3n$ bit is multiplied into the operand D (d_{n-1} … d_1d_0) and the final product would be obtained. Processors perform the above steps in distinct hardware. Dot notation architecture of a 4×4 multiplier is illustrated in Fig. 3.

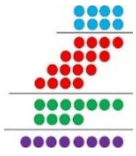


Fig. 3 – Dot notation architecture of a 4×4 multiplication

In order to multiply the third operand C, it is essential to use an 8×4 multiplier (Fig. 4).

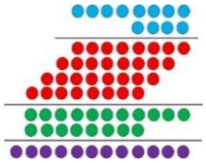


Fig. 4 – Dot notation architecture of an 8×4 multiplication

For a four-bit four-operand multiplication, which is the desired objective, operand D is multiplied into 12 bit result obtained from the previous stage; therefore, the architecture of a 12×4 multiplication would be expressed as Fig. 5.

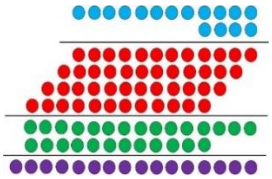


Fig. 5 – Dot notation architecture of a 12×4 multiplication

4.1 Delay Calculation

There exist several methods for implementing the partial product reduction step. In this article, full adders and (4:2) compressors are applied simultaneously which results in decreasing the design cost and speeding up the circuit [2, 19, 20].

Assume that δ is the unit delay in a multiplier architecture; hence, δ_{AND} is AND delay of the first stage of a multiplication (partial product generation). Partial products' height (number of full adders and compressors) is involved in the second step of multiplication. There exist a delay of (CPA) in the final phase. The $T_{n\times n}$ that is the delay of an $n\times n$ multiplication is computed through the following equation:

$$T_{n\times n} = \begin{cases} 7(\lfloor \log_2 n \rfloor - 1) + 4n - 3 & 3 < n \le 5 \\ 7(\lceil \log_2 n \rceil - 1) + 4n - 5 & n > 5 \end{cases} \quad (4\text{-}1)$$

where, δ_{AND} is the delay of an AND gate, n is the partial products' height, $\lfloor \log_2 n \rfloor - 1$ and $\lceil \log_2 n \rceil - 1$ are the number of compressors used for reduction stage, $\delta_{(3:2)}$ is the delay of one stage full adder and $\delta_{CPA(2n-4)}$ and $\delta_{CPA(2n-5)}$ are the latencies of $a\ (2n-4)$ bit and $a\ (2n-5)$ bit CPAs, respectively. Thus, the delay of a 4×4 multiplier is attained through a (4:2) compressor, a full adder and a four-bit CPA. The output of an $n \times n$ multiplication is $2n$ bit that is multiplied into the third operand and will generate a $3n$ bit. A $2n \times n$ bit multiplication delay is expressed by:

$$T_{2n\times n} = \begin{cases} \delta_{AND} + (\lfloor \log_2 n \rfloor - 1)\delta_{(4:2)} + \delta_{(3:2)} + \delta_{CPA_{(3n-4)}} & 3 < n \le 5 \\ \delta_{AND} + (\lceil \log_2 n \rceil - 1)\delta_{(4:2)} + \delta_{(3:2)} + \delta_{CPA_{(3n-5)}} & n > 5 \end{cases} \quad (4\text{-}2)$$

According to equation above, the critical path of an 8×4 multiplier passes through a (4:2) compressor, a full adder and an eight-bit CPA. For multiplying the fourth

operand, a $3n \times n$ bit multiplication is required. The delay of this circuit is calculated through:

$$T_{3n\times n} = \begin{cases} \delta_{AND} + (\lfloor \log_2 n \rfloor - 1)\delta_{(4:2)} + \delta_{(3:2)} + \delta_{CPA_{(4n-4)}} & 3 < n \le 5 \\ \delta_{AND} + (\lceil \log_2 n \rceil - 1)\delta_{(4:2)} + \delta_{(3:2)} + \delta_{CPA_{(4n-5)}} & n > 5 \end{cases} \quad (4\text{-}3)$$

The constituent components of Eq. (4-2) and (4-3) are the same as Eq. (4-1).Therefore, the total delay of a four-operand multiplier is computed by adding the results obtained from above mentioned equations. In order to calculate the delay of final CPAs, the following three equations are introduced [9]:

$$\delta_{CPA_{(n\times n)}} = \begin{cases} \delta_{XOR} + 2(2n-4)\delta_{AND} & 3 < n \le 5 \\ \delta_{XOR} + 2(2n-5)\delta_{AND} & n > 5 \end{cases}$$

$$\delta_{CPA_{(2n\times n)}} = \begin{cases} \delta_{XOR} + 2(3n-4)\delta_{AND} & 3 < n \le 5 \\ \delta_{XOR} + 2(3n-5)\delta_{AND} & n > 5 \end{cases}$$

$$\delta_{CPA_{(3n\times n)}} = \begin{cases} \delta_{XOR} + 2(4n-4)\delta_{AND} & 3 < n \le 5 \\ \delta_{XOR} + 2(4n-5)\delta_{AND} & n > 5 \end{cases}$$

In order to calculate the multiplier delay, with respect to n (number of bits), assume that there is one unit delay for AND/OR gate (δ), 1.5 unit delay for XOR, 3.5 unit delay for a full adder and 7 unit delay for a compressor. By applying values of δ_{CPA} in above mentioned equations, the total delay of $n \times n$, $2n \times n$ and $3n \times n$ multipliers are calculated

5. PROPOSED ARCHITECTURE FOR A FOUR-BIT FOUR-OPERAND MULTIPLICATION

These proposed architectures have a general block diagram that consists of a hardware with four inputs and one output. Here, we emphasize that all the designs are implemented in a single hardware, (see Fig. 6).

Fig. 6 – Four-operand multiplier cell

A four-operand multiplier is presented in [18], which is designed at gate level for which three methods are introduced. In this article an extended version the four operand multiplier is presented emphasizing on theoretical and justification of the achievements of the presented architectures.

5.1 Design I

Here, all three 4×4, 8×4 and 12×4 multiplications are implemented in a unit hardware, where, first, two operands are multiplied and their partial products are generated, next, the reduction stage is carried out through the compressors and full adders, then, the *2n* bit result is acquired through a CPA. The above procedure is carried out for the third and fourth operands. This architecture is presented in Fig. 7. This architecture requires three CPAs. The total delay of an n-bit four-operand multiplier is calculated through adding the results obtained from equations in section four. There exist two cases in this respect:

1) If $3 < n \le 5$ then:

$$T_{n\times n\times n\times n} = 3\delta_{AND} + 3(\lfloor \log_2 n \rfloor - 1)\delta_{(4:2)} + 3\delta_{(3:2)} + \delta_{CPA_{(2n-4)}} + \delta_{CPA_{(3n-4)}} + \delta_{CPA_{(4n-4)}}$$

The term $\delta_{CPA(n\times n\times n\times n)}$ is the result of the sum of the three CPA delays, which is equal to:

$$\delta_{CPA_{(n\times n\times n\times n)}} = 3\delta_{XOR} + (18n-24)\delta_{AND}$$

Hence, the total delay of a four-bit four-operand multiplier with respect to $3 < n \le 5$ is:

$$T_{4\times 4\times 4\times 4} = 3\delta_{AND} + 3\delta_{(4:2)} + 3\delta_{(3:2)} + \delta_{CPA_{(4\times 4\times 4\times 4)}} = 51\delta_{AND} + 3\delta_{XOR} + 3\delta_{(3:2)} + 3\delta_{(4:2)}$$

2) If $n > 5$ then:

$$T_{n\times n\times n\times n} = 3\delta_{AND} + 3(\lceil \log_2 n \rceil - 1)\delta_{(4:2)} + 3\delta_{(3:2)} + \delta_{CPA_{(2n-5)}} + \delta_{CPA_{(3n-5)}} + \delta_{CPA_{(4n-5)}}$$

Consequently, the total delay of a four-bit four-operand multiplier with respect to n>5 is computed through equation:

$$T_{4\times 4\times 4\times 4} = 3\delta_{AND} + 3\delta_{(4:2)} + 3\delta_{(3:2)} + \delta_{CPA_{(n\times n\times n)}} = 45\delta_{AND} + 3\delta_{XOR} + 3\delta_{(3:2)} + 3\delta_{(4:2)}$$

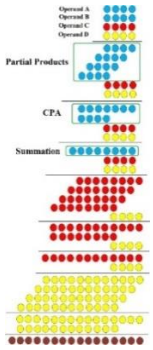
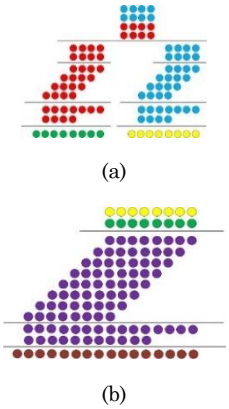


Fig. 7 – Proposed design I for a four operand multiplier

5.2 Design II

Two phases are involved here: The operands *A* and *B* and operands C *and* D are multiplied in pairs, simultaneously, where the partial products are reduced to vectors of *sum* and *carry*. The products of these two multiplications are obtained by applying CPAs in final stages (Fig. 8 (a)). The results of phase one (two 8 bit operands) are going to be multiplied in the next step. The dot notation architecture of this design is shown in Fig. 8 (b). Applying the three CPAs is essential in this design.



(a)

(b)

Fig. 8 – Proposed design II for a four operand multiplier (a) two parallel 4 × 4 multipliers, (b) a 8 × 8 multiplier

The total delay is the delay of two parallel 4×4 multipliers added to an 8×8 multiplier delay expressed as follows:

$$T_{4\times4\times4\times4} = T_{4\times4} + T_{8\times8}$$

The $T_{4\times4}$ is calculated as follow:

$$\begin{aligned} T_{4\times4} &= \delta_{AND} + (\lfloor \log_2 4 \rfloor - 1)\delta_{(4:2)} + \delta_{(3:2)} + \delta_{CPA(2n-4)} \\ &= \delta_{AND} + (\lfloor \log_2 4 \rfloor - 1)\delta_{(4:2)} + \delta_{(3:2)} + (\delta_{XOR} + 2(2\times4-4)\delta_{AND}) \\ &= 9\delta_{AND} + \delta_{XOR} + \delta_{(3:2)} + \delta_{(4:2)} \end{aligned}$$

The $T_{8\times8}$ is also obtained through:

$$\begin{aligned} T_{8\times8} &= \delta_{AND} + (\lceil \log_2 8 \rceil - 1)\delta_{(4:2)} + \delta_{(3:2)} + \delta_{CPA(2n-5)} \\ &= \delta_{AND} + (\lceil \log_2 8 \rceil - 1)\delta_{(4:2)} + \delta_{(3:2)} + (\delta_{XOR} + 2(2\times8-5)\delta_{AND}) \\ &= 23\delta_{AND} + \delta_{XOR} + \delta_{(3:2)} + 2\delta_{(4:2)} \end{aligned}$$

Thus, the total delay of this design is computed by adding above mentioned equations:

$$T_{4\times4\times4\times4} = 32\delta_{AND} + 2\delta_{XOR} + 2\delta_{(3:2)} + 3\delta_{(4:2)}$$

5.3 Design III

Here, the operand A is multiplied into operand B leading to the generation of the partial products. In the next step, these partial products are multiplied into the operand C and generate the new partial products followed by multiplying the operand D into the new partial products, which yields the last partial product. By applying the methods used for partial product reduction, the two vectors of *sum* and *carry* are produced.

In this design, three two-input AND gates are required for partial products generation in three stages. Hence, three unit delay is considered in this phase. The partial products' height is calculated through arithmetic operations which is the number of compressors used in reduction stage, and the result is applied in reduction stage delay. The total delay of an n-bit four-operand multiplication is computed through:

$$T_{n\times n\times n\times n} = 3\delta_{AND} + \left\lfloor \log_2 \left(\frac{2n^3 + n}{3} \right) \right\rfloor \delta_{(4:2)} + \delta_{(3:2)} + \delta_{CPA_{(3n-5)}}$$

By using the above equation and set n equal to 4, the delay of a four-bit four-operand multiplier is obtained through:

$$T_{4\times4\times4\times4} = 17\delta_{AND} + \delta_{XOR} + \delta_{(3:2)} + 6\delta_{(4:2)}$$

The results of the comparisons among these three designs, considering unit delays of each component, is tabulated in Table 1.

Table 1 – Gate level comparison for proposed multipliers

Proposed multiplier	Delay
Design I	81 unit delay
Design II	63 unit delay
Design III	64 unit delay

As observed in Table 1, the total delay of the first design is the worst, due to applying three CPAs in the final step of multiplication. The second method is the best proposed design, regarding delay, due to hardware parallelism. Design III is in between the other two, since one CPA is applied there.

5.4 Implementation of Proposed Multipliers using CNT Technology

The three newly proposed designs are going to be implemented through CNTFETs. To implement the designs of four-bit four-operand multipliers, based on nanotube technology, an AND gate, a full adder cell proposed by the authors, a compressor and a ripple carry adder are applied as the main components (Fig. 9).

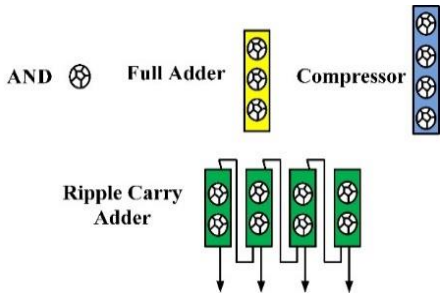


Fig. 9 – Multipliers main components for a CNTFET-based circuit

Transistor implementation of AND gate and the full adder are illustrated in Fig. 10 and 11, respectively.

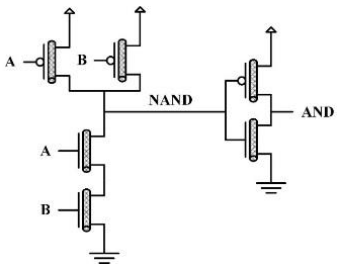


Fig. 10 – A CNTFET-based AND gate

The full adder cell applies a Majority function to produce the C_{out} and XOR-XNOR functions in order to generate the *Sum*. This full adder is composed of 3 capacitors and 14 transistors.

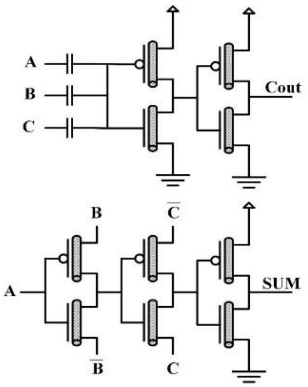


Fig. 11 – A CNTFET-based full adder cell

As observed in Fig. 11, this full adder is constructed through two separate circuits in order to generate *Sum* and C_{out}. To get a full swing output, two inverters are applied to the last stage of the *Sum* and C_{out} in the circuit, which effects the driving capability.

By replacing the Components of Fig. 8 by the components of Fig. 9, the architecture of a CNTFET-based 8×8 multiplier is achieved. This design outperforms its counterparts. The implementation of the other two are carried in the same manner to achieve CNTFET based circuits.

6. SIMULATION RESULTS

The Simulations are conducted through HSPICE simulator. The conventional four-bit four-operand multiplier and these three proposed designs are simulated through CNT technology and three parameters of power, delay and PDP are assessed in the circuits. Power parameter is the average power of the circuit. Delay is calculated from the time when the input signal reaches $1/2V_{dd}$ up to the time when the output signal reaches $1/2V_{dd}$ [21]. The PDP is the product of the average power and delay, representing the trade-off between power and delay in a circuit. For simulating the CNTFET circuits, the spice model proposed in [22-25] is adopted. This standard model is designed for enhancement-mode unipolar MOSFET-like CNTFETs, where each transistor may include more than one CNT as its channel. Moreover, this model takes into account a realistic, circuit-compatible CNTFET structure and includes practical device nonidealities, parasitics, Shottkey-barrier effects at the contacts, doped source-drain extension regions, scattering (nonideal near-ballistic transport), inter-CNT charge screening effects, back-gate (substrate bias) effect and Gate and Source/Drain, resistances and capacitances. The model also comprises a full transcapacitance network for more accurate dynamic and transient performance simulations. The important parameters of the CNTFET model, their corresponding values and a brief description, are tabulated in Table 2.

Table 2 – CNTFET parameters

Parameters	Description	Value
L_{ch}	Physical channel length	32 nm
L_{geff}	Mean free path in the intrinsic CNT channel	100 nm
L_{ss}	Length of doped CNT source-side extension region	32 nm
L_{dd}	Length of doped CNT drain-side extension region	32 nm
K_{gate}	Dielectric constant of high-k top gate dielectric material	16
T_{ox}	Thickness of high-k top gate dielectric material	4 nm
C_{sub}	Coupling capacitance between the channel region and the substrate	40 pF/m
E_{fi}	The Fermi level of the doped S/D tube	6 eV

Simulations of proposed four-bit four-operand multipliers and the conventional counterpart is made at 0.6 and 0.9 voltages and the results are tabulated in Tables 3 and 4.

Table 3 – Simulation results for proposed multipliers at 0.6v, 3fF and 25 °C

Multipliers	Power (e – 06 W)	Delay (e – 10 S)	PDP (e – 16 J)
Conventional four-operand multiplier	14.8582	4.906	72.8943
Proposed design I	18.624	3.8646	71.9743
Proposed design II	12.6542	3.9926	50.5231
Proposed design III	31.185	3.6011	112.3003

With respect to the source voltage of 0.6v and capacitance load of 3fF, the second proposed multiplier has gained 14 % improvement in power consumption in comparison with its conventional counterpart. The delay enhancements in proposed designs I, II and III are 21 %, 18 % and 26 %, respectively, in comparison with the conventional multiplier. The PDP parameter optimization has a slight improvement in the first design, the second has a 30 % optimization and the third is insignificant.

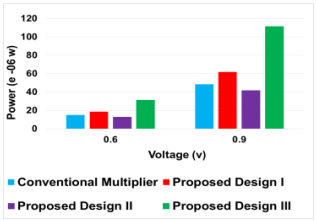
In order to evaluate the performance of the designs, simulations are made at 0.9v, Table 4.

As observed in Table 4, there is no improvement in power consumption in the proposed designs except the second one, which has 13 % optimization at this parameter. The delay of the proposed circuits against the conventional multiplier has an acceptable enhancement of 26 %, 32 % and 16 % in proposed designs I, II and III, respectively. The results of PDP parameter indicate that the first and the second designs have 5 % and 41 % improvement in comparison with the conventional multiplier.

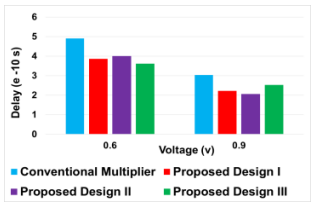
Table 4 – Simulation results for proposed multipliers at 0.9v, 3fF and 25 °C

Multipliers	Power (e – 06 W)	Delay (e – 10 S)	PDP (e – 16 J)
Conventional four-operand multiplier	48.2117	3.0263	145.9030
Proposed design I	61.8140	2.2259	137.5917
Proposed design II	41.7580	2.0492	85.5704
Proposed design III	111.35	2.5182	280.4015

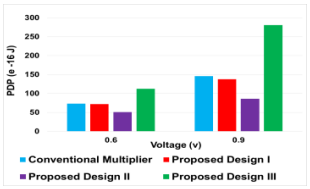
The power, delay and PDP comparisons of the three proposed multipliers and their conventional counterpart are illustrated in Fig. 12.



(a)



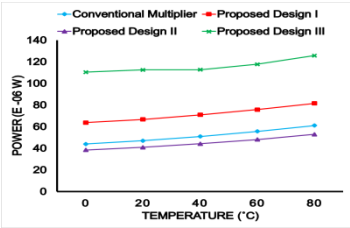
(b)



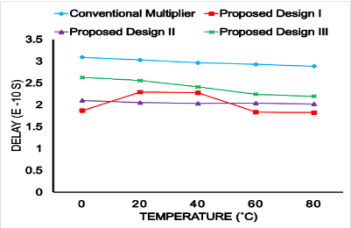
(c)

Fig. 12 – (a) Power, (b) delay, (c) PDP versus different voltages in multipliers

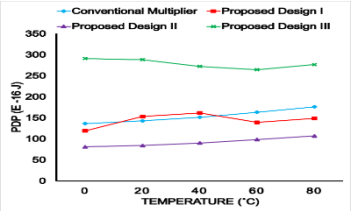
Considering another important aspect of noise immunity, the values of PDP for proposed multipliers and the conventional counterpart at different temperatures are simulated. Each circuit is considered under a vast range of temperatures, from 0°C to 80°C, to evaluate the sensitivity of each design to temperature variation. Fig. 13 illustrates the results of this experiment at supply voltage of 0.9v and load capacitance of 3fF.



(a)



(b)



(c)

Fig. 13 – (a) Power, (b) delay, (c) PDP versus different temperatures in multipliers

It can be inferred from Fig. 13 that the proposed design II is the superior design, can perform in a vast range of temperatures and has an acceptable functionality in the parameters of the power, delay and PDP compared with other designs.

7. CONCLUSION

In this article, three new architectures for a four-bit four-operand are introduced. These proposed multipliers are designed and implemented through CNT technology and compared with their conventional counterpart. In order to implement the multipliers in this article, a full adder cell is applied based on CNTFET technology. The first proposed multiplier multiplies four operands in a unit hardware, which leads to a slight improvement in delay and PDP parameters. The second design applies a parallel multiplication, resulting in a reduction in power consumption and increase in speed; hence, a reduction in PDP parameter in comparison with its conventional multiplier. The third design applies one CPA in the last stage of multiplication; therefore a notable decrease in delay parameter is observed.

REFERENCES

1. J.Y. Kang, J.L. Gaudiot, *Euromicro Symposium on Digital System Design* 508 (2004).
2. N. Itoh, Y. Naemura, H. Makino, Y. Nakase, T. Yoshihara, Y. Horiba, *IEEE J. Solid-State Circuits* **36**, 249 (2001).
3. S. Mhaske, I. Ghosekar, P. Bhaskar, *Int. J. Eng. Res. General Sci.* (2015).
4. W.C. Yeh, C.W. Jen, *IEEE T. Comput.* **49** No 7, 692 (2000).
5. B. Parhami, *Computer arithmetic: algorithms and hardware designs* (Oxford University Press: 2009).
6. S. Murugeswari, S.K. Mohideen, *Mobile Communication and Power Engineering* (Springer Berlin Heidelberg: 2013).
7. M.R. Reshadinezhad, K. Navi, *Int. J. Comput. Sci. Network* (2012).
8. M.R. Reshadinezhad, N. Charmchi, K. Navi, *Int. J. Modern Education Comput. Sci.* **7**, 44 (2015).
9. G. Cho, Y.B. Kim, F. Lombardi, M. Choi, *IEEE International Instrumentation and Measurement Technology Conference* 909 (2009).
10. M.R. Reshadinezhad, M.H. Moaiyeri, K. Navi, *IEICE Transaction. Electron.* **95**, 744 (2012).
11. S. Iijima, *Nature* **354**, 56 (1991).
12. Y. Maheswar, B.L. Raju, D.K.S. Rajan, *Int. J. Sci. Eng. Res.* **4** (2011).
13. P.L. McEuen, M.S. Fuhrer, H. Park, *IEEE T. Nanotechnol.* **1**, 78 (2002).

14. S.A. Ebrahimi, M.R. Reshadinezhad, A. Bohlooli, M. Shahsavari, *Microelectron. J.* **53**, 156 (2016).
15. S.E. Abed, B.J. Mohd, Z. Al-bayati, S. Alouneh, *Int. J. Circuit Theory Applications* **40**, 1175 (2012).
16. R.S. Waters, E.E. Swartzlander, *IEEE T. Comput.* **59**, 1134 (2010).
17. N. Charmchi, M.R. Reshadinezhad, *Int. J. Adv. Information Sci. Technol.* **43**, 14 (2015).
18. M.R. Reshadinezhad, N. Charmchi, M. Masoud, *7th international conferance on Nanotechnology (ICN-2017)* (Tbilisi: Georgia: 2017).
19. A. Pishvaie, G. Jaberipur, A. Jahanian, *Canad. J. Electrical Comput. Eng.* **36**, 111 (2013).
20. C.H. Chang, J. Gu, M. Zhang, *IEEE T. Circuits System.* **51**, 1985 (2004).
21. H.E. Neil, D. Harriss, *CMOS VLSI Design, Thired Edition* (Addison Wesley Publishiong: 2005).
22. J. Deng, H.P. Wong, *IEEE T. Electron Dev.* **54**, 3186 (2007).
23. J. Deng, H.P. Wong, *IEEE T. Electron Dev.* **54**, 3195 (2007).
24. J. Deng, H.P. Wong, *International Conference on Simulation of Semiconductor Processes and Devices* 166 (2006).
25. *Stanford University. 2014. Stanford CNTFET Model.* [Online] Available at: https://nano.stanford.edu/stanford-CNTFET-model