Received March 16, 2019, accepted April 1, 2019, date of current version April 22, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2909500*

# Ternary Functions Design Using Memristive Threshold Logic

**NANCY SOLIMAN[1], MOHAMMED E. FOUDA[2,3], ABDULLAH G. ALHARBI[4], (Member, IEEE), LOBNA A. SAID[1], AHMED H. MADIAN[1,5], (Senior Member, IEEE), AND AHMED G. RADWAN[2,6], (Senior Member, IEEE)**

[1]Nanoelectronics Integrated Systems Center, Nile University, Cairo 12588, Egypt
[2]Engineering Mathematics and Physics Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt
[3]Electrical Engineering and Computer Science Department, University of California Irvine, Irvine, CA 92697, USA
[4]Department of Electrical Engineering, Faculty of Engineering, Jouf University, Sakaka 72388, Saudi Arabia
[5]Radiation Engineering Department, NCRRT, Egyptian Atomic Energy Authority, Cairo 29 SOS, Egypt
[6]School of Engineering and Applied Sciences, Nile University, Cairo 12588, Egypt

Corresponding author: Mohammed E. Fouda (foudam@uci.edu)

**ABSTRACT** Memristive threshold logic (MTL) concept is emerged in many circuits to enable high-performance systems in terms of power, energy, area, and delay. This paper proposes a systematic method for building two-bit ternary number functions based on the MTL concept. The proposed method is applied to build the basic ternary arithmetic operations. The implementation of two-bit adder and multiplier is presented in the unbalanced ternary number representation. The proposed designs are verified by using VTEAM memristor and Stanford CNTFET transistor models. Finally, a comparison between the proposed circuits and related work presented in this paper is discussed. It shows that the area in case of the ternary adder is reduced by 30% and 76% and in case of the ternary multiplier by considering that memristors can be stacked above the transistors. In addition, this reduction in the number of transistors reduces the circuit static power and hence improving the overall ternary circuits performance.

**INDEX TERMS** Ternary numbers, threshold logic, mermistor, VTEAM, CNTFET, STI, PTI, NTI, SBI, STB, SBB, MTL, TLG.

## I. INTRODUCTION

Multi-level electronic systems which offer a prominent reduction in implementation' complexity, power consumption, and area, were used in digital application to build multi-number base systems [1]. Ternary systems are one of these recent systems that provide more information to be represented in the same number of digits compared to the binary systems [1]. In addition, its odd valued logic facilities representing rational and complex number rather than binary numbers [2]. These superior properties are stimulated to improve the computational performance in the fields of artificial intelligence logic [3], cryptography systems [4] and fuzzy logic systems [5]. In the 1950s, it was the first attempt to build ternary emulator computer [6]. Later, researchers moved from emulators design to implement the first arithmetic operations based on different technologies such as CMOS, carbon nano-tube (CNTFET) transistors, and different integrated technologies [7]–[11], [22].

Since Chua postulated the memristor in 1971, it was considered an alternative to the transistor [12], [13]. It has not only a small area but also non-volatile multilevel storage with fast switching speed [14]. This behavior inspires the researchers to define a recent concept called memristor threshold logic (MTL) which resembles synaptic of neurons in the brain [15], [16]. It is composed of two parts; one part works as operational synaptic circuit and the other works as control decision circuit which fires synaptic circuit output [15]. It has opened the space for speeding up and minimizing the area of many potential systems [15], [17]. So, it has emerged in many applications such as; logic gates, fast switching memory, pattern recognition, image processing, and biological brain neuron network [18], [19].

Unlike binary number systems, the two-bit ternary number can represent upto 729 different functions while the binary system only represents 16 functions. In this paper, a novel

The associate editor coordinating the review of this manuscript and approving it for publication was Stavros Souravlas.

and systematic method is presented to design and realize any two-bit unbalanced ternary function based on the concept of memristor threshold logic (MTL). In this work, we focus on the realization on the logic gate and circuit implementations. the theoretic analysis is beyond the scope of this paper. The proposed method is applied to build a two-bit ternary adder and multiplier with different designs based on memristive logic gates. The memristor specifications required for proper work is presented. The simulation result and a comparison between the proposed arithmetic circuits and related work are introduced.

This paper is organized as follows: Section II introduces a summary of ternary logic gates (TLGs). Section III introduces ternary logic voltage ranges and the flow of the proposed method. Section IV presents the designs of two-bit ternary adder and multiplier as examples for employing the proposed method. Section V presents the simulation results and comparison with previous works. Finally, the conclusion is drawn in section VI.

## II. TERNARY LOGIC GATES

There are two main representations of the ternary numbers [23]; one is the balanced representation where a $-1$, 0, and 1 voltage levels are used to represent the ternary numbers and the other one is the unbalanced representation where 0, 1 and 2 voltage levels are used. In this work, we focus on the unbalanced representation.

### A. TERNARY INVERTERS AND BUFFERS

There are three types of the ternary inverters; simple ternary inverter (*STI*), positive and negative ternary inverters (*PTI* and *NTI*). The input and output of *STI* inverter take three possible voltage levels which describe ternary numbers. On the other hand, *PTI* and *NTI* outputs take only two voltage levels, either high or low. In case of *NTI*, the input middle voltage level goes to the low output voltage (0). However, for *PTI*, the input middle level voltage goes to high output voltage ($V_{dd}$).

**TABLE 1.** Ternary inverters functionality.

| Unbalanced | | | |
|---|---|---|---|
| $x$ | $STI(x)$ | $PTI(x)$ | $NTI(x)$ |
| 0 | 2 | 2 | 2 |
| 1 | 1 | 2 | 0 |
| 2 | 0 | 0 | 0 |

In this paper, the following abbreviations are considered; *SBI* is referred to the standard binary inverter, *SBB* is referred to the usual binary buffer and *STB* is referred to simple ternary buffer. Table 1 shows the truth table of ternary inverters in the unbalanced ternary representation. Figure 1 depicts the circuit implementations of the ternary inverters and buffers with their symbols.

The *PTI*, *NTI* and *SBI* have the same circuit structure. However, the difference in their behaviors is due to the transistors' dimensions ($d$). For instance, when $d_{T_1} = d_{T_2}$,

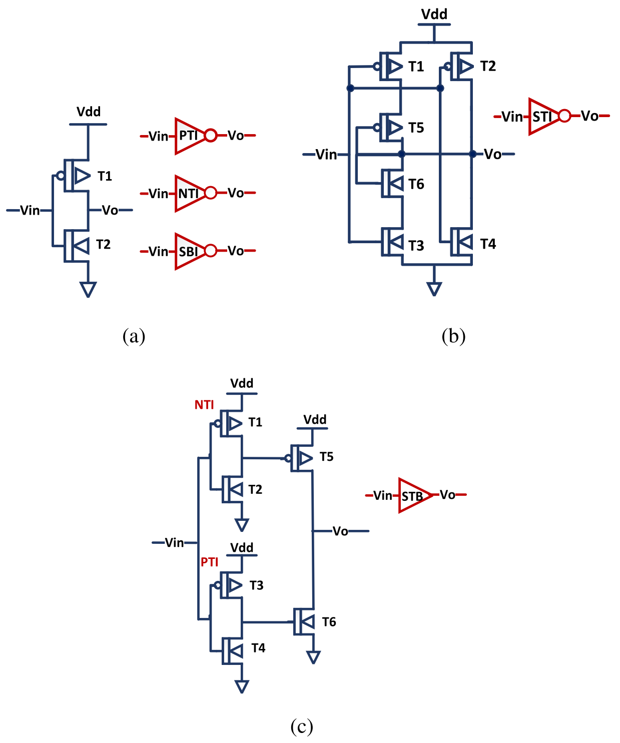


**FIGURE 1.** Schematic of (a) standard binary inverter or positive ternary or negative ternary inverter, (b) standard ternary inverter, and (c) standard inverter buffer circuits.

*SBI* is obtained. However, when $d_{T_1} > d_{T_2}$, *PTI* is obtained and vice versa for *NTI* inverter. The behaviors of the ternary inverters and buffer circuits are explained and discussed in detail in [24], [25].

### B. MEMRISTIVE TOR, TAND GATES AND T-AVG CIRCUIT

*TOR* and *TAND* are basic ternary logic gates (*TLG*) where their outputs are determined by the maximum and minimum input voltage as described by (1a) and (1b), respectively. Ternary voltage averaging circuit (*TAvg*) does not belong to the *TLG*s. However, it is emerged in the applications based on the threshold voltage concept [15]. The output of *TAvg* is determined by the average sum of the applied input voltage as described by (1c).

$$TOR(v_1, v_2, ..., v_n) = Max[v_1, v_2, ..., v_n], \tag{1a}$$
$$TAND(v_1, v_2, ..., v_n) = Min[v_1, v_2, ..., v_n], \tag{1b}$$
$$TAvg(v_1, v_2, ..., v_n) = (v_1 + v_2 + ... + v_n)/n, \tag{1c}$$

*TOR*, *TAND*, and *TAvg* were previously implemented by different technologies [20], [21], [26], [27]. Figures 2 (a-c) show different configurations of two series switching memristors which represent the implementations of two-bit *TOR*, *TAND* and *TAvg*, respectively [20], [21], [26], [27]. Their memristances switch on and off ($R_{on}$ and $R_{off}$) according to the input voltages. Table 2 summarizes memristances states and the output corresponding to different possible cases of the applied input in each circuit. The current direction flow

**TABLE 2.** TOR and TAND circuits different cases.

| Cases | $TOR$ | $TAND$ | $TAvg$ |
|---|---|---|---|
| $v_{in_1} = v_{in_2}$ | No current flow | No current flow | No current flow |
| | $v_o = v_{in_{1,2}}$ | $v_o = v_{in_{1,2}}$ | $v_o = v_{in_{1,2}}$ |
| $v_{in_1} > v_{in_2}$ | $v_{m_1} < 0$ $R_{m_1} = R_{on}$ | $v_{m_1} > 0$ $R_{m_1} = R_{off}$ | $v_{m_1} < 0$ $R_{m_1} = R_{on}$ |
| | $v_{m_2} > 0$ $R_{m_2} = R_{off}$ | $v_{m_2} < 0$ $R_{m_2} = R_{on}$ | $v_{m_2} < 0$ $R_{m_2} = R_{on}$ |
| | $v_o = v_{in_1}$ | $v_o = v_{in_2}$ | $v_o = 0.5(v_{in_1} + v_{in_2})$ |
| $v_{in_1} < v_{in_2}$ | $v_{m_1} > 0$ $R_{m_1} = R_{off}$ | $v_{m_1} < 0$ $R_{m_1} = R_{on}$ | $v_{m_1} > 0$ $R_{m_1} = R_{off}$ |
| | $v_{m_2} < 0$ $R_{m_2} = R_{on}$ | $v_{m_2} > 0$ $R_{m_2} = R_{off}$ | $v_{m_2} > 0$ $R_{m_2} = R_{off}$ |
| | $v_o = v_{in_2}$ | $v_o = v_{in_1}$ | $v_o = 0.5(v_{in_1} + v_{in_2})$ |
| Memrsitance behavior | $R_{on}$ is obtained by the maximum input and $R_{off}$ is obtained by the minimum input. | $R_{off}$ is obtained by the maximum input and $R_{on}$ is obtained by the minimum input. | Memristances increase to $R_{off}$ or decrease to $R_{on}$, simultaneously. |

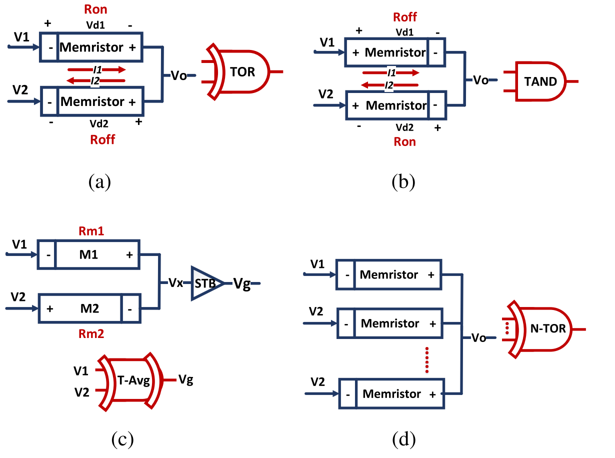


**FIGURE 2.** Memristive implementation of two-bit (a) *TOR*, (b) *TAND*, (c) T-Avg, and (d)N-bits TOR.

through the memristor sets the polarity of the voltage across the memristance ($v_m$) which determines the memristance state as shown in Table 2. For instance, when the current flows from the negative to the positive of memristor, $v_m < 0v$ and the memristance decreases to $R_{on}$ and vice-versa for the different polarity as shown in Fig.2 (a-c). As the outputs of $TOR$ and $TAND$ tend to the input corresponding to the minimum memristance. In the case of *TAvg*, the output is always the average of the inputs. Figure 2(d) shows the implementation of $N$- bits $TOR$, $TAND$ and $TAvg$ but with different memristors polarities as depicted in Fig.2 (b-c).

## III. THE PROPOSED MTL DESIGN METHODOLOGY FOR TWO-BIT TERNARY FUNCTIONS

### A. TERNARY LOGIC VOLTAGE RANGE IDENTIFICATION

*MTL* design concept is essentially based on the threshold voltage. So, it is needed to identify precisely the voltage range for each ternary logic level. Table 3 shows the distributed voltage ranges for each logic level in ternary number unbalanced representation. By considering these voltage logic identifications, *TAvg* function can be represented as shown in Table 3 according to the logic inputs values ($A$ and $B$). It shows also the ternary logic output of $TOR$, $TAND$ functions described by (1).

**TABLE 3.** Ternary logic voltage levels and logic gates output.

| | logic 0 | logic 1 | Logic 2 |
|---|---|---|---|
| Voltage Range | $[0, 0.3V_{dd}]$ | $[0.35V_{dd}, 0.65V_{dd}]$ | $[0.7V_{dd}, V_{dd}]$ |
| $TOR$ | (0,0) | (0,1) (1,1) | (0,2), (1,2), (2,2) |
| $TAND$ | (0,0), (0,1), (0,2) | (1,1), (1,2) | (2,2) |
| $TAvg$ | (0,0), (0,1) | (0,2), (1,1) | (1,2) (2,2) |

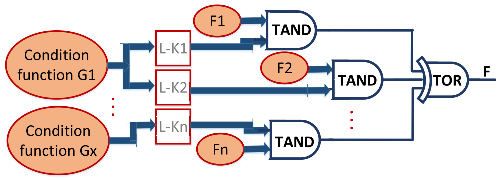


**FIGURE 3.** MTL design general architecture.

### B. MAIN IDEA OF THE PROPOSED MTL DESIGN METHOD

The proposed threshold Logic (TL) design method is built based on splitting the ternary function into small sub-functions. Each sub-function is represented by a separable path in the circuit design. Each path is assigned by ON state (short circuit) under specific ternary logic condition while the others are assigned by OFF state (open circuit). This idea is so close to the definition of decoding where according to a certain logic input, only one path in a circuit is enabled and the others are disabled. Figure 3 shows the proposed general architecture for ternary functions where the function $f$ is divided into $n$ sub-functions ($f_1, f_2$, ... and $f_n$). Each sub-function path is enabled according to a certain condition determined by the two blocks ($G$ and $L - k$). $G$ block is a condition function built from basic ternary logic gates (TOR and TAND). $L - k$ (ternary logic checker) is a comparator to check on a certain logic, outputs from the condition function $G$. There are three types of $L - k$: one for check logic 0, the second to check logic 1 and the other to check logic 2. The output of each type becomes either logic 2 or 0 based on the input either it matches the comparator type or

**TABLE 4. MTL logic shifters and condition functions.**

| Logic Shifter | | |
|---|---|---|
| $0 \longrightarrow 1$ | $2 \longrightarrow 0$ | $0 \longrightarrow 2$ |
| $TAND(STI(x),1)$ <br> $TAND(PTI(x),1)$ <br> or <br> $TAND(NTI(x),1)$ | $NTI(x)$, <br> $PTI(x)$ <br> or <br> $SBI(x)$ | $NTI(x)$, <br> $PTI(x)$ <br> or <br> $SBI(x)$ |
| $1 \longrightarrow 0$ | $1 \longrightarrow 2$ | $2 \longrightarrow 1$ |
| $NTI(x)$ | $PTI(x)$ or $SBI(\,NTI(x)\,)$ | $TAND(x,1)$ |
| Logic Conditions | | |
| $if(x == 0)$ | $if(x == 1)$ | $if(x == 2)$ |
| $NTI(x)$ | $TAND(\,SBI(\,NTI(x)\,),PTI(x)\,)$ | $SBI(\,PTI(x)\,)$ |

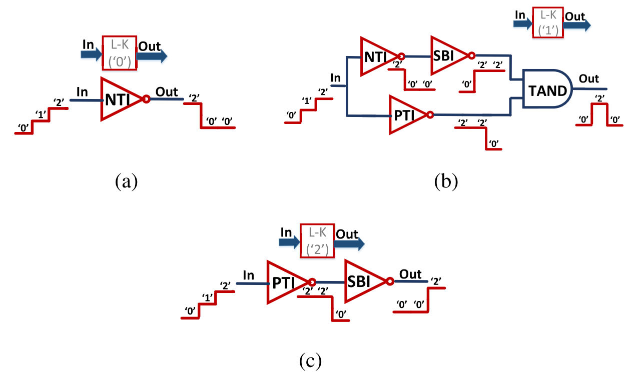


**FIGURE 4. Ternary logic checkers $L-K$ (a) Logic 0, (b) Logic 1, and (c) Logic 2.**

not, respectively. For instance, if logic 1 is assigned as input to $L-k$ of logic 1 type, the output is logic 2 and otherwise, the output is logic 0. Figure 4 shows the proposed implementations of each $L-k$ type where NTI is used to check the logic 0. SBI with PTI is used to check logic 2 and logic 1 is checked by ANDing the combined inverters PTI, SBI, and NTI.

## C. THE PROPOSED MTL DESIGN METHOD

The main principle in the proposed design method is to generate a pseudo-code describing the ternary logic function then synthesizing this code into the ternary gate level.

The general design flow is summarized in the following steps:

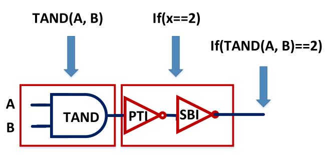
- Divide the required function into sub-functions ($f_1, f_2, .., f_n$).
- Multiplex between each part by a specific condition.
- Write a pseudo-code describing the function based on sub-functions and their multiplier conditions.
- Synthesis this pseudo code to ternary logic gates and inverters.

### 1) THE FOLLOWING STEPS EXPLAIN THE PROPOSED MTL METHODOLOGY:

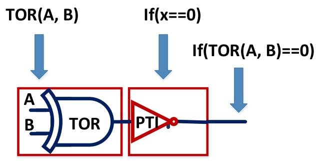
1) Group all $f$ inputs that have the same outputs.
2) Select any of the possible input pairs (points) that belongs to either the group $f = 1$ or $f = 2$ in *step* 1. The selected one is considered as an initial point to determine one of the $f$ sub functions.
3) Define a special condition for one of the sub functions ($f_x$) as follows:
   a) Select one of the three gates (*TAND*, *TOR*, and *TAvg*).
   b) Apply selected initial point as an input to this logic gate.
4) Determine all other possible inputs which meet the condition in *step* 3.
5) Obtain the outputs of $f_x$ for the inputs in *step* 4.
6) Compare the outputs of three logic gates with the outputs of $f$ in *step* 5. As a result, the possible cases can be summarized as follows:
   - *Matched*: $f_x$ can be described using the matched logic gate under the condition defined in *step* 3.
   - *Mismatched*: Use the logic shifter functions shown in Table 4 to convert mismatched logic outputs to its corresponding logic output in *step* 5.
     – If shifter functions for all mismatched outputs are the same: go to *step* 7.
     – If shifter functions are different: add another condition gate stage to eliminate the mismatched outputs.
7) Write a pseudo code of the sub-function $f_x$.
8) Remove all points of $f_x$ from function $f$. Then, follow the same procedure starting from *step* 2 on the remained points to determine the other sub-functions of the main function.
9) Implement the pseudo-code as follows:
   - Implement each condition by the functions described in Table 4 as the example shown in Fig.5.
   - ANDing each part of the sub-functions with the corresponding condition as the example shown in Fig.6(a)
   - ORing all sub functions outputs to get the full function as the exampel shown in Fig.6(b).

## IV. BUILDING TERNARY LOGIC BLOCKS BASED ON THE PROPOSED METHOD

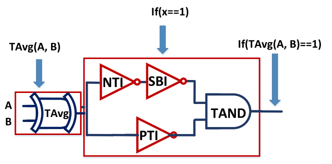
The proposed method has been employed to build different two-bit ternary logic functions in order to minimize area,



(a)

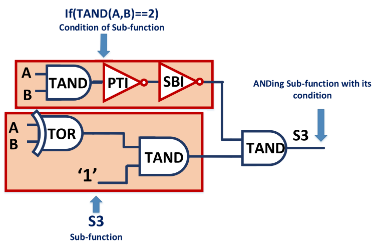


(b)

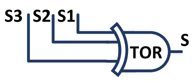


(c)

**FIGURE 5. Condition implementation examples (a) *TAND(A, B)* == 2, (b) *TOR(A, B)* == 0, and (c) *TAvg(A, B)* == 1.**


If(TAND(A,B)==2)
Condition of Sub-function
A
B
TAND
PTI
SBI
ANDing Sub-function with its condition
A
B
TOR
TAND
'1'
TAND
S3
S3
Sub-function

(a)



(b)

**FIGURE 6. Implementation steps (a) ANDing sub function with its condition and (b) ORing all sub function of sum.**

power, energy, and delay. This method builds pseudo-code which describes the logic function. Then this deduced code is synthesized to ternary logic gates circuit. Finally, these logic circuits down convert from logic circuits to memristor and transistor level.

**TABLE 5. Two bit ternary adder truth table.**

| | *Logic* 0 | *Logic* 1 | *Logic* 2 |
|---|---|---|---|
| $S(A,B)$ | $(0,0),(1,2)$ | $(0,1),(2,2)$ | $(0,2),(1,1)$ |
| $C(A,B)$ | $(0,0),(0,1),(0,2),(1,1)$ | $(1,2),(2,2)$ | $-$ |

## *A. TWO-BIT TERNARY ADDER*

The proposed method has been employed to build two-bit ternary adder described by the functions $S$ and $C$ in Table 5. $S$ function is divided into three sub-functions $S_1$, $S_2$ and $S_3$ and the method steps have been applied for each part as shown in Table 6 as follows:

1) For $S = 0, 1$ and $2$ , the inputs groups are [(0, 0),(1, 2)], [(0, 1),(2, 2)] and [(0, 2),(1, 1)], respectively.

2) Select one random point in the groups $S = 1$ or $S = 2$. In this example (0, 1) point has been taken as an initial point, then refer this sub function to $S_1$ .

3) To define a special condition for this sub-function ($S_1$):

   a) *TAND* is select from the possible logic gates realizations (*TAND*, *TOR*, *TAvg*).

   b) Initial point selected in *step* 2 is applied to this logic gate as follows $TAND(0, 1) = 0$.

   From this step, the condition of $S_1$ is $TAND(A, B) == 0$.

4) According to the condition in *step* 3, the other possible pair inputs that met this condition is (0, 0) and (0, 2). Consequently, $S_1$ is represented by the inputs [(0, 0), (0, 1) and (0, 2)].

5) According to the inputs [(0, 0), (0, 1) and (0, 2)] which represents $S_1$, the outputs of $S_1$ are 0, 1 and 2, respectively.

6) For the inputs in *step* 4 [(0, 0), (0, 1) and (0, 2)], compare each logic gate outputs with $S$ outputs in *step* 5. As shown in Table 3, $S_1$ outputs match the outputs of TOR logic gate. Therefore, $S_1$ can be described by TOR logic gate.

7) $S_1$ pseudo-code can be described as follows: "When $TAND(A, B)= 0$, the output $S_1 = TOR(A, B)$".

8) Remove input pairs [(0, 0), (0, 1) and (0, 2)] from the inputs in *step* 1.

   Then, go back to *step* 2 to start realizing the second part $S_2$ with the remaining possible inputs [(1, 2),(2, 2) and (1, 1)] with the same procedure. By following the same steps again:

   - The point (1, 1) is selected randomly and *TAND* is selected as a logic gate condition $S_2$.
   - The condition of $S_2$ is $TAND(A, B) == 1$. The other possible inputs which met $S_2$ condition are (2, 1) and (1, 2).
   - Hence, $S_2$ is represented by the inputs [(1, 1), (2, 1), (1, 2)].
   - According to these inputs, $S_2$ outputs are 2 and 0, respectively.
   - By comparing these outputs with the outputs of three logic gates (*TAND*, *TOR* and *TAvg*), its found that there is no output matches $S_2$ output when the inputs are [(1, 1), (2, 1), (1, 2)] as shown in Table 3. Therefore, Go to shift logic Table 4 to match the output of $S_2$ to one of logic gates (*TAND*, *TOR*, *TAvg*). Shift TOR logic outputs as follows:
     - When input is (1, 1), TOR output is 1. This output needed to be shifted to 2 to match the output of $S_2$. This shift can be done by using the function $PTI(TOR(A, B))$ where A and B are the input to TOR.
     - When input is (2, 1) or (1, 2), TOR output is 2. These output needed to be shifted to 0 to match the output of $S_2$. These shift can be done by using also the function $PTI(TOR(A, B))$.

**TABLE 6.** **MTL method steps to build sum function of two bit ternary adder.**

| # | $S_1$ | $S_2$ | $S_3$ |
|---|---|---|---|
| 1 | $S = 1$ @ $(0,1),(2,2)$ <br> $S = 2$ @ $(0,2),(1,1)$ | $S = 1$ @ $(2,2)$ <br> $S = 2$ @ $(1,1)$ | $S = 1$ @ $(2,2)$ <br> $S = 2$ @ $-$ |
| 2 | Initial point: <br> $(0,1)$ @ $S = 1$ | $(1,1)$ @ $S = 2$ | $(2,2)$ @ $S = 1$ |
| 3 | Condition gate: $TAND$ <br> Where $TAND(0,1) = 0$, <br> Condition: <br> $TAND(A,B) == 0$ | Condition gate: $TAND$ <br> Where $TAND(1,1) = 1$, <br> Condition: <br> $TAND(A,B) == 1$ | Condition gate: $TAND$ <br> Where $TAND(2,2) = 2$, <br> Condition: <br> $TAND(A,B) == 2$ |
| 4 | $TAND(A,B) = 0$ @ <br> $(0,0),(0,1),(0,2)$ | $TAND(A,B) = 1$ @ <br> $(1,1),(1,2)$ | $TAND(A,B) = 2$ @ <br> $(2,2)$ |
| 5 | @ $(0,0) \longrightarrow S = 0$ <br> @ $(0,1) \longrightarrow S = 1$ <br> @ $(0,2) \longrightarrow S = 2$ | @ $(1,1) \longrightarrow S = 1$ <br> @ $(1,2) \longrightarrow S = 0$ | @ $(2,2) \longrightarrow S = 1$ |
| 6 | $TOR$ outputs match <br> $S$ output on *step* 5 | No gate outputs matches $S$ output on *step* 5. Hence, <br> Go to logic shifter table | |
| 7 | $If(TAND(A,B){==}0$: <br> $S_1 = TOR(A,B)$ | $If(TAND(A,B) == 1)$ : <br> $S_2 =$ <br> $PTI(TOR(A,B))$ | $If(TAND(A,B) == 2)$ : <br> $S_3 =$ <br> $TAND(TOR(A,B),1)$ |
| 8 | $S = TOR(S_0, S_1, S_2)$ | | |

**Procedure 1** Pseudo Code for two-bit Ternary Adder Using TAND Gate as Condition

```
if $TAND(A, B) == 0$ then
    $S_1 = TOR(A, B)$
    $S_2 = S_3 = 0$
else if $TAND(A, B) == 1$ then
    $S_2 = PTI(TOR(A, B))$
    $S_1 = S_3 = 0$
else if $TAND(A, B) == 2$ then
    $S_3 = TAND(TOR(A, B), 1)$
    $S_1 = S_2 = 0$
end if
$S = TOR(S_1, S_2, S_3)$
if $TOR(A, B) == 1$ then
    $C = TAND(TAND(A, B), '1')$
else
    $C = 0$
end if
```

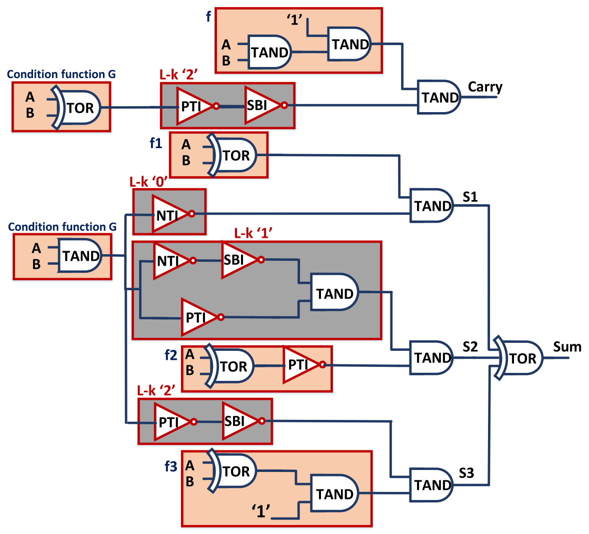


**FIGURE 7.** **Two-bit ternary adder MTL circuit design 1.**

Consequently, the deduced $S_2$ pseudo-code is described as follows: when $TAND(A, B) = 1$, the output $S_2 = PTI(TOR(A, B))$.

- By removing points (1, 1) and (1, 2) from the previous remained inputs [(1, 2),(2, 2), and (1, 1)], the point (2, 2) is remained to represent the third $S$ sub function ($S_3$). By following the same procedure, its function can be deduced to $TAND(TOR(A, B), '1')$ as shown in Table 6.

9) Each sub-function pseudo-code has been synthesized to ternary logic gates, inverters as shown in Fig.7 as follows:
   - Conditions have been implemented as described in Table 4.
   - Each condition is ANDed with its corresponding sub-function.
   - Finally, the ANDs' outputs are ORed together to get the full implementation of $S$ function.

For the $C$ function, similarly, the MTL design steps has been followed with condition $TOR(A, B) == 1$ and described by the function $TAND(TAND(A, B), '1')$. Then, it has synthesized to ternary logic gates and inverters as shown in Fig.7. List 1 shows $S$ and $C$ functions the pseudo-code.

In general, the proposed *MTL* method provides different designs for the same function because of design variances in *steps* 2, 3 and 6. Figure 8 shows another design for two-bit ternary adder using MTL method. *TAvg* logic gate and *TAND* are selected as gate conditions for both the $S$, and the $C$ functions as shown from the pseudo-code in list 2. The function $S$ is divided into three sub-functions as follows: $S_1$ which represents points [(0, 2), (1, 1)] by condition $TAvg(A, B) = 1$, always has logic 2 which considers the

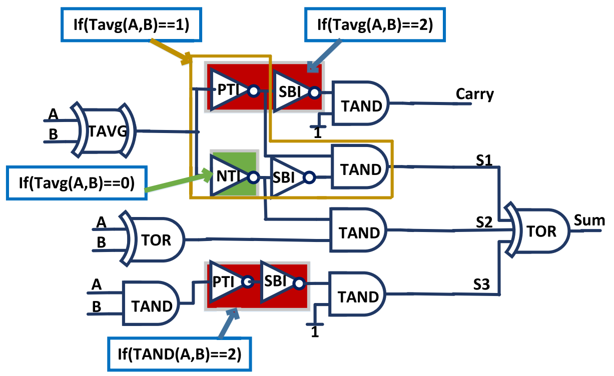


**FIGURE 8.** Two-bit ternary adder MTL circuit design 2.

output of the condition itself directly. $S_2$ outputs which represents points of [(0, 0), (0, 1)] by condition $TAvg(A, B) = 0$, can be described by logic gate TOR. $S_3$ which represents point (2, 2) by condition $TAND(A, B) = 2$, always has logic 1 and can be described by $TAND(TAND(A, B), 1)$. Similarly, for the $C$ function which always has logic 1, the points [(1, 2),(2, 2)], can be described by $TAND(TAvg(A, B), 1)$ under condition $TAvg(A, B) = 2$. Finally, Fig.8 shows the corresponding synthesized logic of the pseudo code of the two-bit adder.

**Procedure 2** Pseudo-code for two-bit Ternary Adder Using TAvg Gate as Condition

```
if $TAvg(A, B) == 1$ then
    $S_1 = 2$
    $S_2 = S_3 = 0$
else if $TAvg(A, B) == 0$ then
    $S_2 = TOR(A, B)$
    $S_1 = S_3 = 0$
else if $TAvg(A, B) == 1$ then
    $S_3 = 1$
    $S_1 = S_2 = 0$
end if
$S = TOR(S_1, S_2, S_3)$
if $TAvg(A, B) == 2$ then
    $C = 1$
else
    $C = 0$
end if
```

### B. TWO-BIT TERNARY MULTIPLIER

The proposed method has been employed also to build two-bit ternary multiplier described by the functions $P$ and $C$ in Table 7. The same steps, explained in Table 6, have been followed. $TAND$ is selected as logic gate condition in both parts $P_1$ and $P_2$ as described in list 3 as follows: $P_1$ represents point (2, 2) by condition $TAND(A, B) = 2$, always has logic 1 and can be described by $TAND(TAND(A, B), 1)$. $P_2$ which represents points [(1, 1), (1, 2)] by condition $TAND(A, B) = 1$, can be described by $TOR(A, B)$ logic gate. For the $C$ function which always has logic 1 for the point (2, 2), can be described by the same function of $P_1$. Finally, the pseudo-code described in list 3 is synthesized to design shown in Fig.9

**TABLE 7.** Two-bit ternary multiplier truth table based on *P* and *C* functions.

| | Logic 0 | Logic 1 | Logic 2 |
|---|---|---|---|
| $P(A,B)$ | (0,0), (0,1), (0,2) | (1,1), (2,2) | (1,2) |
| $C(A,B)$ | (0,0), (0,1), (0,2), (1,1), (1,2) | (2,2) | – |

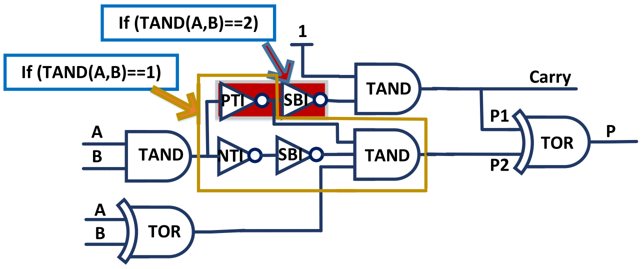


**FIGURE 9.** Two-bit ternary multiplier circuit design.

**Procedure 3** Pseudo Code for two-bit Ternary Multiplier Using TAND Gate as Condition

```
if $TAND(A, B) == 2$ then
    $P_1 = C = 1$
    $P_2 = 0$
else if $TAND(A, B) == 1$ then
    $P_2 = TOR(A, B)$
    $P_1 = C = 0$
else
    $P_1 = P_2 = C = 0$
end if
$P = TOR(P_1, P_2)$
```

## V. SIMULATION RESULTS AND COMPARISON

The proposed method designs are verified using bipolar switching memristor in addition to carbon nanotube (CNTFET) transistors [28], [30]. Figure 10 shows the simulation results of ternary inverters and buffer using CNTFET transistors for the circuits shown in Fig.1. The transistor model of Stanford Virtual-Source Carbon Nanotube (VS CNTFET) is used in simulation [29]. Figure 11 shows the simulation results of ternary number logic gates using memrsitors for the circuits shown in Fig.2. In general, there are critical conditions on memristor parameters for the memristive logic gates circuits to work properly during transistor and memristor integration [10], [11]. The memristor switching time has to be bounded through memristive logic gate pole location at $f_p = 1/(2\pi R_{on} C_{in_{tr}})$. Hence, the time constant is $tau = R_{on} C_{in_{tr}}$. To give the signal a time to settle (reach steady state) $t_{op_{mem}} \leq 5\tau$ which means that

$$f_{op_{Max_{mem}}} = \frac{1}{2\pi R_{on} C_{in_{tr}}}. \quad (2)$$

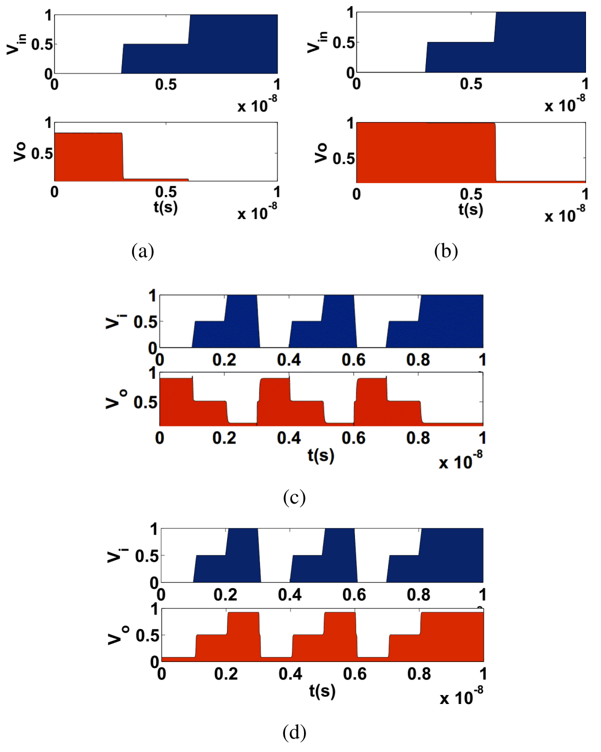


**FIGURE 10.** **The simulation results of (a) *NTI*, (b) *PTI*, (c) *STI*, and (d) *STB*.**

To integrate transistor with memristor, the operating frequency of the transistor and memristor have to be compatible. Therefore, the $f_{op_{tr}} = f_{op_{mem}}$ which configures the memristor $R_{on}$ parameter through (2). As $R_{off} \gg R_{on}$ (about 10 times) [13], the $f_{op_{tr}}$ configures also the memristor $R_{off}$ parameter. Once the $R_{on}$ and $R_{off}$ are configured, the memristor parameters $v_{th_1}$ and $v_{th_2}$ can be configured as follows:

$$v_{th_1} = -v_{th_2} = \frac{R_{on} V_{dd}}{2(R_{on} + R_{off})}, \tag{3}$$

where $v_{th_1}$ and $v_{th_2}$ are the minimum voltage thersholds need to change the memrsitance from $R_{on}$ to $R_{off}$ and vice versa.

For VS-CNTFET transistor model that is used in our simulations, the $C_{inCNT} = 4.83 \times 10^{-15}$ and $t_{op} = 1nsec$ [30]. Consequently, the appropriate memristor model has to work on $f_{op} = 1GHz$, hence, $R_{onMax} = 6.59k\Omega$. If $R_{on} = 50\Omega$ is selected , the $R_{off} = 1k\Omega$ and the $v_{th_1} = -v_{th_2} = 0.023v$ which satisfied with the parameters of the VTEAM memristor model reported in [30].

Figure 12 shows the simulation results of the proposed ternary $L - K$ logic checkers circuits described in Fig.4. Each $L - K$ type has been verified using VTEAM memristor and CNTFET transistor where $A_0$, $A_1$, and $A_2$ represent the outputs of each $L - K$ logic 0, 1, 2 checkers, respectively. Figure 13 shows the simulation results of the proposed ternary adders and multipler. Table 8 shows CNTFET dimensions used in simulation.

Table 9 shows a comparison between the proposed designs based on the MTL method and other related

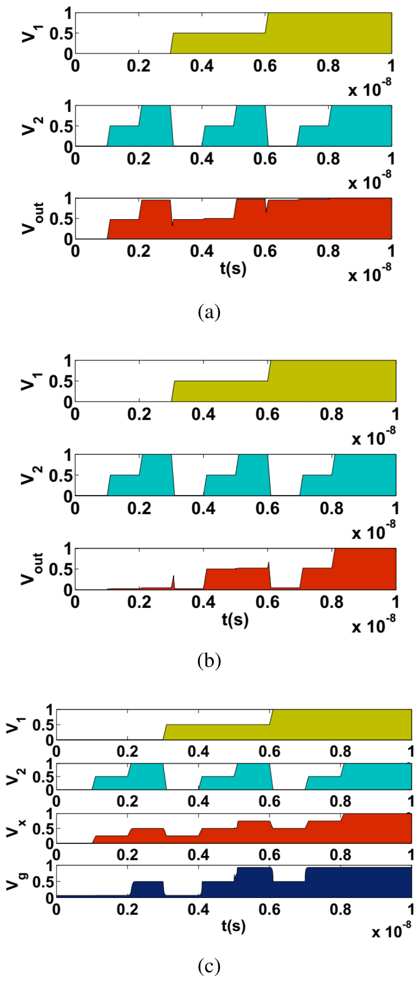


**FIGURE 11.** **The simulation results of (a) *TOR*, (b) *TAND*, and (c) *TAvg* ternary memristive gates.**

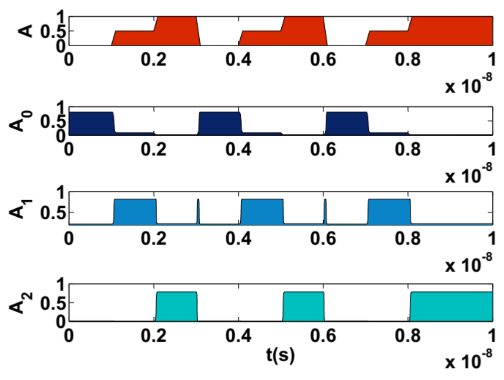


**FIGURE 12.** **The simulation results of the memristive- CNTFET $L - K$ logic checker.**

work [20], [21], [31]. Table 10 illustrates the longest path delay of the proposed designs where $D_{TAND_2}$ is the delay of a two-bit memristive *TAND* logic gate, $D_{TAND_3}$ is the delay of a two-bit memristive *TAND* logic gate and so on. As result, the proposed method assists to reduce the number of transistors and memristors in the design of the ternary function.

**TABLE 8.** CNTFET transistors tubes' diameters.

|  | $STI$ |  |  |  |  |  | $NTI$ |  | $PTI$ |  |
|---|---|---|---|---|---|---|---|---|---|---|
| Inverters $(nm)$ | $d_{T_1}$ | $d_{T_2}$ | $d_{T_3}$ | $d_{T_4}$ | $d_{T_5}$ | $d_{T_6}$ | $d_{T_1}$ | $d_{T_2}$ | $d_{T_1}$ | $d_{T_2}$ |
|  | 2 | 1.3 | 2 | 1.3 | 1 | 1 | 1.3 | 2 | 2 | 1.3 |
|  | $STB$ |  |  |  |  |  | $SBB$(two series $SBI$) |  |  |  |
| Buffers $(nm)$ | $d_{T_1}$ | $d_{T_2}$ | $d_{T_3}$ | $d_{T_4}$ | $d_{T_5}$ | $d_{T_6}$ | $d_{T_1}$ | $d_{T_2}$ | $d_{T_3}$ | $d_{T_4}$ |
|  | 1.3 | 2 | 2 | 1.3 | 2 | 2 | 2 | 2 | 2 | 2 |

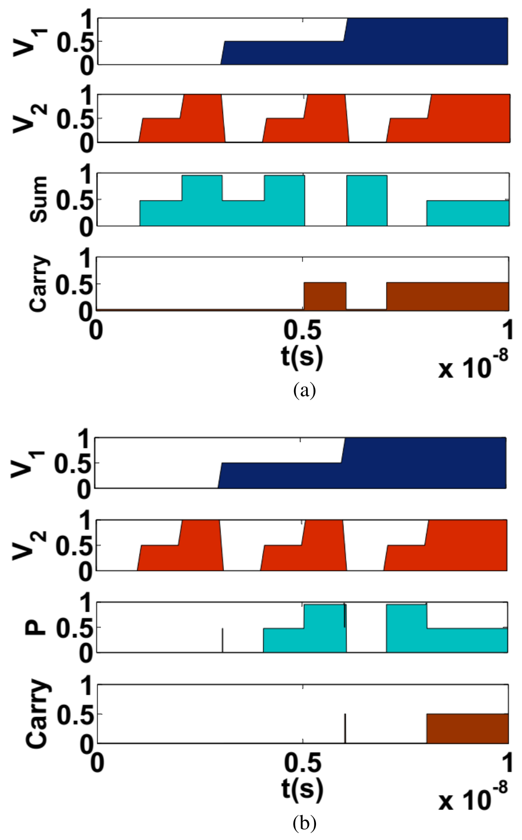


**FIGURE 13.** The simulation results of two-bit (a) T-Adder and (b) T-Multiplier.

**TABLE 9.** Two-bit ternary adder and multiplier comparison.

|  | Designs | # memristors | # transistors |
|---|---|---|---|
| T-Adder | K-Map CNTFET | – | ∼ 138 |
|  | K-Map memristor | 35 | 56 |
|  | MTL Design 1 | 17 | 40 |
|  | MTL Design 2 | 17 | 34 |
| T-Multiplier | K-map CNTFET | – | ∼ 108 |
|  | MTL Design | 11 | 26 |
| Reduced Area | T- Adder |  | T-Multiplier |
|  | 30% |  | 76% |

**TABLE 10.** Two-bit ternary adder and multiplier proposed circuit delay.

|  |  | Path Delay |
|---|---|---|
| T_Adder | Design 1 | $D_{TAND_2} + D_{TAND_3} + D_{TOR_3} + 2D_{SBI} + 2D_{STB}$ |
|  | Design 2 | $D_{TAvg_2}+ D_{TAND_2}+ D_{TOR_3}+3D_{STB}+2D_{SBI}$ |
| T- Multiplier |  | $D_{TAND_2}+ D_{TAND_3}+ 2D_{SBI} +D_{TOR_2} +D_{STB}$ |

Hence, reducing the overall area and static power of ternary circuits. Therefore, the proposed method is very promising to be emerged to build high-performance ternary computing units and other applications such as fuzzy logic.

## VI. CONCLUSION

A general design method was introduced to build two-bit ternary numbers functions based on the MTL concept. MTL designs were built based on the basic memristive ternary logic gates and employed ternary inverters as threshold logic comparators (checkers). The proposed method is used to build basic ternary arithmetic operations in unbalanced representation. The proposed designs were verified using VTEAM memristor and CNTFET transistor. A comparison with other related work was held on and showed a reduction in the number of transistors and memristor. Consequently, reducing the overall area and static power of ternary circuits. The proposed method is very promising to be emerged to build high-performance ternary computing units and other applications such as fuzzy logic. The proposed method can be easily automated into computer-aided design tool to generate the ternary functions.

The main limitation of the proposed implementation is the requirements on the set and reset voltages of the devices which are not satisfied in all memristor devices. Also, the proposed technique requires devices with high endurance since the devices are switching with changing the inputs.

## REFERENCES

[1] V. Dimitrov, G. Jullien, and R. Muscedere, *Multiple-Base Number System: Theory and Applications*. Boca Raton, FL, USA: CRC Press, 2017.

[2] M. M. A. Taha and M. Perkowski, "Realization of arithmetic operators based on stochastic number frequency signal representation," in *Proc. IEEE 48th Int. Symp. Multiple-Valued Logic (ISMVL)*, May 2018, pp. 215–220.

[3] R. Caferra, "Artificial Intelligence," *Logic for Computer Science and Artificial Intelligence*. Hoboken, NJ, USA: Wiley, 2013, pp. 245–257.

[4] J. Adikari, V. S. Dimitrov, and L. Imbert, "Hybrid binary-ternary number system for elliptic curve cryptosystems," *IEEE Trans. Comput.*, vol. 60, no. 2, pp. 254–265, Feb. 2011.

[5] T. J. Ross, "Logic and fuzzy systems," *Fuzzy Logic with Engineering Applications*. Hoboken, NJ, USA: Wiley, 2010, pp. 117–173.

[6] J. Burroni, "The act of computer programming in science," in *Proc. Int. Conf. Art, Sci., Eng. Program.*, 2017, Art. no. 30.

[7] A. Srivastava and K. Venkatapathy, "Design and implementation of a low power ternary full adder," *VLSI Des.*, vol. 4, no. 1, pp. 75–81, 1996.

[8] PH. D. Chung-Yu Wu and H.-Y. Huang, "Design and application of pipelined dynamic CMOS ternary logic and simple ternary differential logic," *IEEE J. Solid-State Circuits*, vol. 28, no. 8, pp. 895–906, Aug. 1993.

[9] J. Knoch and J. Appenzeller, "Electronic transport in carbon nanotube field-effect transistors," *Molecular- and Nano-Tubes*. Boston, MA, USA: Springer, 2011, pp. 355–389.

[10] T. F. Wu *et al.*, "Brain-inspired computing exploiting carbon nanotube FETs and resistive RAM: Hyperdimensional computing case study," in *ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 492–494.

[11] M. M. Shulaker *et al.*, "Monolithic 3D integration of logic and memory: Carbon nanotube FETs, resistive RAM, and silicon FETs," in *IEDM Tech. Dig.*, Dec. 2014, pp. 27.4.1–27.4.4.
[12] R. S. Williams, "Aftermath of finding the memristor," *Memristor Networks*. Cham, Switzerland: Springer, 2014, pp. 15–19.
[13] A. G. Radwan and M. E. Fouda, *On the Mathematical Modeling of Memristor, Memcapacitor, and Meminductor* (Studies in Systems, Decision and Control). Cham, Switzerland: Springer, 2015.
[14] A. P. James, "The memristor circuits and applications," in *Advances in Memristor Circuits and Bioinspired Systems*. Singapore: Research Publishing, 2015.
[15] A. K. Maan, D. A. Jayadevi, and A. P. James, "A survey of memristive threshold logic circuits," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1734–1746, Aug. 2017.
[16] R. Kozma, R. E. Pino, and G. E. Pazienza, Eds., *Advances in Neuromorphic Memristor Science and Applications*. Cham, Switzerland: Springer, 2012.
[17] P. Mazumder, S. M. Kang, and R. Waser, "Memristors: Devices, models, and applications," *Proc. IEEE*, vol. 100, no. 6, pp. 1911–1919, Jun. 2012.
[18] A. P. James, D. S. Kumar, and A. Ajayan, "Threshold logic computing: Memristive-CMOS circuits for fast Fourier transform and vedic multiplication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2690–2694, Nov. 2015.
[19] A. K. Maan, D. S. Kumar, S. Sugathan, and A. P. James, "Memristive threshold logic circuit design of fast moving object detection," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2337–2341, Oct. 2015.
[20] N. S. Soliman, M. E. Fouda, and A. G. Radwan, "Memristor-CNTFET based ternary logic gates," *Microelectron. J.*, vol. 72, pp. 74–85, Feb. 2018.
[21] N. S. Soliman, M. E. Fouda, L. Said, A. Madian, and A. G. Radwan, "Memristor-CNTFET based ternary comparator unit," in *Proc. 30th Int. Conf. Microelectron. (ICM)*, 2018.
[22] A. A. El-Slehdar, A. H. Fouad, and A. G. Radwan, "Memristor-based balanced ternary adder," in *Proc. 25th Int. Conf. Microelectron. (ICM)*, 2013, pp. 1–4.
[23] G. Hang, Y. Yang, D. Zhang, and X. Hu, "Dynamic ternary logic gate using neuron-MOS literal circuit and double pass-transistor logic," in *Proc. 12th Int. Conf. Comput. Intell. Secur. (CIS)*, Dec. 2016, pp. 82–86.
[24] M. H. Moaiyeri, A. Doostaregan, and K. Navi, "Design of energy-efficient and robust ternary circuits for nanotechnology," *IET Circuits, Devices Syst.*, vol. 5, no. 4, p. 285, 2011.
[25] S. Lin, Y.-B. Kim, and F. Lombardi, "A novel CNTFET-based ternary logic gate design," in *Proc. 52nd IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2009, pp. 435–438.
[26] S. H. Amer, A. H. Madian, H. ElSayed, A. S. Emara, and H. H. Amer, "Theory, modeling and design of memristor-based min-max circuits," in *Advances in Memristors, Memristive Devices and Systems* (Studies in Computational Intelligence). Cham, Switzerland: Springer, 2017, pp. 187–205.
[27] T. Breuer, L. Nielen, B. Roesgen, R. Waser, V. Rana, and E. Linn, "Realization of minimum and maximum gate function in $Ta_2O_5$-based memristive devices," *Sci. Rep.*, vol. 6, no. 1, Apr. 2016, Art. no. 23967.
[28] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: Threshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Jan. 2013.
[29] *Stanford University CNFET Model*. Accessed: Mar. 2018. [Online]. Available: https://nano.stanford.edu/stanford-cnfet-model
[30] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.
[31] S. Lin, Y. B. Kim, and F. Lombardi, "CNTFET-based design of ternary logic gates and arithmetic circuits," *IEEE Trans. Nanotechnol.*, vol. 10, no. 2, pp. 217–225, Mar. 2011.

**NANCY SOLIMAN** received the B.Sc. degree in electronics and communications engineering from the Faculty of Engineering, Cairo University, Giza, Egypt, in 2016. She is currently pursuing the M.Sc. degree. She is also a Research Assistant with the Nanoelectronics Integrated System Center (NISC), Nile University, Giza. She has authored over seven journal and conference papers. Her research interests include digital IC design, RTL, FPGA, electronic design automation tool, machine learning, algorithms, digital design formal verification, and embedded systems. She has served as a Reviewer for ICM 2018 in Tunisia.

**MOHAMMED E. FOUDA** received the B.Sc. degree (Hons.) in electronics and communications engineering and the M.Sc. degree in engineering mathematics from the Faculty of Engineering, Cairo University, Cairo, Egypt, in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the University of California Irvine, USA. His Ph.D. research is focused on enabling the emerging devices (Memristor/RRAM) for memory and neuromorphic applications. He has authored or coauthored over 55 journal and conference papers. His research interests include brain-inspired computing, neuromorphic circuits and systems, resistive memories, modeling and analysis of mem-elements-based circuits, fractional-order circuits, and analog mixed circuits. He has served as a Technical Program Committee Member of the IEEE International Conference on Microelectronics (ICM) 2018 and the IEEE International Conference on Design and Test Integrated Micro and Nano-Systems (DTS), in 2019. He also serves as a Peer-Reviewer for many prestigious journals and conferences. He has received the Best Paper Award from ICM 2013 and the Broadcom Foundation Fellowship, from 2016 to 2017.

**ABDULLAH G. ALHARBI** (S'10–M'17) received the B.Sc. degree in electronics and communications engineering from Qassim University, Saudi Arabia, in 2010, and the master's and Ph.D. degrees in electrical engineering from the University of Missouri, Kansas City, USA, in 2014 and 2017, respectively. From 2010 to 2012, he was an Electrical Engineer with Saudi Aramco. He is currently an Assistant Professor with the Electrical Engineering Department, Jouf University, Saudi Arabia. He has authored or coauthored over 25 journal and conference papers and one Springer Book Chapter. His research interests include digital IC design, memristor-based circuits, memory devices, and nanoelectronics. He was a recipient of several awards from the Saudi Arabian Cultural Mission, USA, and the University of Missouri, Kansas City. He is a member of the Gulf Engineering Union, the IEEE Circuits and Systems Society, the IEEE Young Professionals, the IEEE Signal Processing Society, the IEEE Instrumentation and Measurement Society, and the IEEE Electron Devices Society. He is also a Professional Member of the ACM.

**LOBNA A. SAID** is a full-time Assistant Professor with the Faculty of Engineering and Applied Science, Nile University. She is the Head of fractional-order circuits and systems research track of the NISC Research Center. She has an H-index ten as reported by Scopus with total citations of 311. She has over 53 publications distributed between high-impact journals, conferences, and book chapters. She was involved in many research grants as a Senior Researcher from different national organizations. Her research interests are interdisciplinary, including system modeling, control techniques, optimization techniques, analog and digital integrated circuits, fractional order circuits and systems, non-linear analysis, and chaos theory. She has received the Recognized Reviewer Award from many international journals. She is the Vice-Chair of research activities at the IEEE Computational Intelligence Egypt Chapter. She has received the Excellence Award from the Center for the Development of Higher education and Research, in 2016.

**AHMED H. MADIAN** (SM'12) was born in 1975. He received the B.Sc. degree (Hons.), and the M.Sc. and the Ph.D. degrees in electronics and communications from Cairo University, Cairo, Egypt, in 1997, 2001, and 2007, respectively. He has been the Micro-Electronics System Design Maser Program Director (MSD) and the Nano-Electronics Integrated System Research Center Director (NISC), Nile University, since 2015. He was a Visiting Associate Professor of electronics with German University in Cairo (GUC), from 2008 to 2013, and an Adjunct Associate Professor with The American University in Cairo (AUC). He is also a member of the National Committee for Radio Science (URSI), Academy of Scientific Research and Technology (ASRT). He has coauthored over 100 research papers in different scientific journals and international conferences. He is currently an Associate Professor with the Electronics Engineering Department, Micro-Electronics Design Center, Egyptian Atomic Energy Authority, Cairo. His research interests include circuit theory, low-voltage analog CMOS circuit design, current-mode analog signal processing, digital VLSI, system security, and mixed/digital applications on field programmable gate arrays, fractional systems, and memristor. He has served as the Program and the Publication Chair for many international conferences and participated in the organization committee for many international conferences. He has received funds for many research projects from different organizations. He is the Co-Founder of the IEEE Robotics Chapter-Egypt Section (Best Chapter on Region 8, in 2013). He has received the Best Researcher Award from Nile University, in 2017.

**AHMED G. RADWAN** (SM'12) was the Former Director of the Nanoelectronics Integrated Systems Center (NISC), Nile University, and the Technical Center for Career Development (TCCD), Cairo University. He was a Visiting Professor with the Computational Electromagnetic Laboratory (CEL), Electrical and Computer Engineering Department (ECE), McMaster University, Canada, from 2008 to 2009. He was with the King Abdullah University of Science and Technology (KAUST), from 2009 to 2011. He has over 250 papers, h-index 35, and over 3500 citations based on Scopus database. He is the Co-Inventor of six U.S. patents. He has authored or coauthored over several international books and 26-chapter books. He has received many research grants as a Principle Investigator (PI), the CO-PI, or a Consultant from different national/international organizations. His research interests include interdisciplinary concepts between mathematics and engineering applications such as fractional-order systems, bifurcation, chaos, memristor, and encryption. Recently, he has selected as an MC Observer to COST Action CA15225, and a member of the Applied Science Research Council, Specialized Scientific Councils (SSC), and the National Committee of Mathematics, ASRT, Egypt. Previously, he was selected as a member of the first scientific council of the Egyptian Young Academy of Sciences (EYAS) and the first scientific council of the Egyptian Center for the Advancement of Science, Technology, and Innovation (ECASTI) to empower and encourage Egyptian young scientists in science and technology and build knowledge-based societies. He has received the State Achievements Award for research in mathematical sciences, in 2012, the Cairo University Achievements Award for research in engineering sciences, in 2013, the Abdul Hameed Shoman Award for Arab Researchers in basic sciences, in 2015, the Cairo University Excellence Award for research in engineering sciences, in 2016, and the Best Researcher Awards from Nile University, in 2015 and 2016. He has organized many special sessions, and he serves on the Technical Program Committee (TPC) of several international conferences. He was invited to be a Lead/Guest Editor of different prestigious international journals.