Implementation and Applications of a Ternary Threshold Logic Gate

Ahmet Unutulmaz and Cem Unsalan
Faculty of Engineering, Department of Electrical and Electronics Engineering, Marmara University, Istanbul, Turkey.

*Corresponding author(s). E-mail(s):
ahmet.unutulmaz@marmara.edu.tr;
Contributing authors: cem.unsalan@marmara.edu.tr;

Abstract
Reducing delay, power consumption, and chip area of a logic circuit are the main targets of a designer. Most of the times, the designer sacrifices power consumption and chip area to improve delay for a given technology node. To overcome this problem, we propose a ternary threshold logic gate. We implement the proposed gate by combining threshold logic and ternary logic. Then, we construct basic building blocks of a ternary ALU (as logic gates, comparator, and arithmetic circuits) using the proposed gate. We show that the proposed ternary TLG improves delay, power consumption, and chip area of ternary circuits via simulations. Thus, the proposed gate can be used to improve delay, power consumption, and chip area of ternary circuits.

Keywords: differential threshold logic, ternary logic, multi-threshold design

1 Introduction
Decreasing delay of a logic circuit is one of the main targets of a designer. Most of the times, the designer sacrifices power consumption to improve delay for a given technology node. One way to overcome this trade-off is the utilization of binary threshold logic gates (TLG). Previous studies showed that delay and power consumption of binary logic circuits can be improved by utilizing TLG [1, 2]. We provide a brief review of threshold logic circuits in Section 2.1. The
designer should also keep track of the design cost in addition to improving delay and power consumption. The cost may be reduced by using less chip area. One way of doing this is via reducing wire (routing) congestion [3]. Ternary logic helps in this respect. A wire carrying a ternary signal (trit) can have three different logic levels, whereas a wire carrying a binary signal (bit) can only have two different logic levels. Therefore, information is spatially compressed in the ternary case. Thus, a ternary logic circuit requires less routing resources and chip area.

Researchers implemented ternary logic circuits via MOSFET and carbon nano-tube FET (CNTFET) technologies. Ternary circuit implementations with complementary MOSFETs (CMOS) dates back to 1980s. Then, the available processes were only offering depletion and enhancement type devices, where ternary circuit implementations were requiring negative potential [4]. For several decades, not much progress has been made in ternary logic implementations. We believe that this was mainly due to the limitations of older implementation technologies. Research in ternary logic gained momentum with the development of CNTFETs [5, 6]. However, practicality of large-scale CNTFET circuits is hard to achieve despite its inherent support for the multi threshold design paradigm. To the authors knowledge, there are no CNTFET implementations of large scale ternary logic circuits. In terms of binary logic, the largest fabricated CNTFET circuit contains only about 15,000 CNTFETs [7]. This is far behind today’s binary designs which utilizes modern MOSFET/FinFET technologies. Although, MOSFET technologies of 1980s were not very suitable for ternary logic circuits, recent MOSFET/FinFET technologies offer devices with different thresholds, even for 7 nm node and beyond [8]. These multi-Vt devices provide an opportunity to decrease power consumption. They may also be used to implement ternary logic circuits. We provide a short overview of ternary logic implementations in Section 2.2.

In this study, we aim to decrease delay, power consumption, and chip area of CMOS ternary circuits. We achieve these by combining advantages of threshold and ternary logic. Therefore, we propose a ternary threshold logic gate (TLG). The proposed ternary TLG is based on modern multi-Vt MOSFET technology. To our knowledge, this is the first study implementing ternary TLG using multi-Vt MOSFETs. We present the ternary TLG in Section 3. We then use it to implement basic blocks of an ALU. Therefore, we first implement the ternary inverter (STI), AND, OR, and XOR gates in Section 4. Next, we implement a ternary comparator in Section 5. Then, we present ternary arithmetic circuits in Section 6. In Section 7, we compare the proposed ternary TLG based circuits with their CMOS only implementations and the ones in literature. Comparison results indicate that the proposed TLG based STI, XOR, comparator, and arithmetic circuits have significant advantages over their CMOS only implementations. Finally, we report key findings and future research directions in Section 8.

1 A depletion mode n-type device is on when gate to source voltage $V_{gs}$ is 0 V where an enhancement mode n-type device is off when $V_{gs} = 0$ V.
2 Background

The proposed ternary TLG is based on two main concepts: threshold logic and ternary logic. We summarize the previous work on these two concepts in this section. We start with the threshold logic and continue with ternary logic.

2.1 Review of Threshold Logic

History of threshold logic and detailed review of its implementations are presented in [9]. There, implementations are categorized into five groups as CMOS solutions, capacitive implementations, conductance/current implementations, single electron tunneling solutions, and resonant tunneling devices. Differential solutions, a sub class of conductance/current implementations, turns out to be the most promising ones in terms of delay and power consumption. These are also known as differential threshold logic gates and abbreviated as DTL gates, differential TLG or simply TLG.

Differential TLG is suitable to be used with standard CMOS logic. Studies have shown that using threshold logic with standard CMOS gates reduces power consumption and necessary chip area [1, 2]. In [1], a 32-bit two’s complement integer multiplier was designed using TLG in conjunction with CMOS standard cells on a 65 nm process node. Compared to CMOS only implementation, the design was 1.23 times smaller, consumed 1.41 times less dynamic power, and had 2.5 times less leakage power. The authors also fabricated their design. In a recent study [2], a standard cell library including TLG was built. Here, an SR latch is abutted after the TLG. The library is then used to synthesize a Wallace tree multiplier, FIR filter, floating point multiplier, 32-bit MIPS core, and AES implementation. Synthesized circuits, including both CMOS and TLG cells, outperformed CMOS only implementations in terms of chip area usage, dynamic and leakage power consumption. A review of synthesis methods targeting threshold logic, as well as a logic synthesis flow which maps a digital circuit function to a threshold logic network, is presented in [10]. The optimization method used in this study was able to estimate the chip area using different cost functions.

Popularity of threshold logic increased with emerging technologies such as memristors [11]. A comprehensive survey on memristive threshold logic circuits was presented in [12]. In [13], a TLG implementation is demonstrated. The authors claimed that utilization of a memristor network via TLG implementation improves robustness. Similarly, memristors were integrated into TLGs in [14] to increase the robustness of logic gates and improve power consumption. Recently, physical implementation of a memristor-based TLG is introduced in [15]. Here, in-house fabricated metal-oxide memristors were used in implementation. The authors demonstrated two, three, and four input TLGs.
2.2 Review of Ternary Logic

First attempts to use multi-valued logic, specifically ternary logic, in computation systems have started several decades ago. The authors [4] stated the shortcomings of previous (T-gate) approaches as their exponential complexity with the number of inputs. They utilized CMOS technology to implement ternary inverters, NAND, and NOR gates using enhancement and depletion mode transistors together. Then, they implemented a full adder using conventional binary logic with ternary-to-binary decoder/encoder circuits before/after the binary circuit. There, calculations were done in binary logic. Input/output signals were encoded in ternary logic. This improved connectivity and reduced number of input and output pins.

Multi-valued logic systems, specifically ternary logic, gained popularity with the development of CNTFETs. This is mainly due to the fact that threshold voltage of a CNTFET can be adjusted by changing its physical dimensions (more precisely, the chirality vector). Transistors with different threshold values will turn on/off at different voltage levels. This significantly eases implementation of multi-valued logic circuits. CNTFETs were used to design ternary inverter, NAND, and NOR gates in [5]. These gates were then combined with binary gates to implement a half adder and one-bit multiplier. CNTFET based half and full-adders and a 3-trit multiplier are presented in [16]. The authors aimed to reduce the number of CNTFETs in [17]. They applied their method on a half and full adder circuit in addition to a 3-trit multiplier. A decoder based high-performance half adder was presented in [18]. Memristor based two-bit ternary adder and multiplier circuits were implemented using CNTFETs in [19, 20]. In these studies, ternary outputs are obtained via memristor based gates. Voltage at the output of these gates were digitized using ternary inverters. The proposed methodology yielded compact circuits. However, inherent static power dissipation of memristor gates may limit their application. A ternary ALU design was presented in [6]. Here, the authors made use of 2-to-1 multiplexers to convert binary and ternary logic instead of using a decoder/encoder based design. The multiplexer-based design methodology improved power consumption compared to encoder/decoder based designs in simulations. Multiplexer based adder and subtractor circuits were implemented in [21]. A ternary logic synthesizer targeting emerging device technologies was presented in [22]. The proposed method was based on a geometric (cubic) representation. The proposed synthesizer runs in two steps. First, the ternary function is expressed in terms of unary functions. Then, the implementation is optimized considering transistor count. Even though not based on ternary logic, a worth mentioning study is [7]. The authors fabricated a binary RISC-V microprocessor with CNTFETs. To our knowledge, this is the largest CNTFET design containing approximately 15,000 CNTFETs. This study shows that CNTFET technology is promising. However, it also indicates that CNTFET technology is far behind advanced MOSFET/FinFET technology nodes which yield designs with several billions of transistors on a single die.
3 Ternary Threshold Logic Gate

Several studies have shown that replacing compatible parts of binary logic with TLGs reduces delay [1, 2, 9]. This is done without sacrificing the power consumption. Or the power consumption is reduced this way without increasing the delay. We extend this idea from binary to ternary logic in this section. Hence, we propose the ternary TLG. To our knowledge, this is the first study implementing ternary TLG using a multi-Vt CMOS process. This gate can be used to decrease delay and power consumption of ternary logic circuits. In this section, we first provide functional form of the proposed ternary TLG with its block diagram. Then, we provide its circuit implementation.

3.1 Ternary TLG Function

Block diagram of the proposed ternary TLG is given in Fig. 1(a). Inputs \((a, b, c, d)\) are in ternary logic in this diagram. Outputs \(o\) and \(\bar{o}\) are in binary form and they are complement of each other. Note that, the binary value 0 (false) corresponds to ternary value 0. Whereas, the binary value 1 (true) corresponds to ternary value 2. Thus, outputs \(o\) and \(\bar{o}\) can take ternary values 0 or 2.

We can formulate the output \(o\) as function of its inputs as

\[ o = \text{sgn}(n_1 \cdot a + n_2 \cdot b + \varepsilon - n_3 \cdot c - n_4 \cdot d) + 1 \]  

where \(\varepsilon\) is a small positive offset, \(n_i\)s are integer coefficients, and \(\text{sgn}(\cdot)\) is the sign function defined as

\[
\text{sgn}(x) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{if } x = 0 \\
-1 & \text{if } x < 0 
\end{cases}
\]  

We will refer the proposed ternary TLG via the symbol in Fig. 1(b) in the following sections.
3.2 Circuit Implementation

Before dealing with circuit implementation, we provide details of the utilized technology. The proposed implementations are done using extended versions of ptm-32nm technology models [23, 24]. Low threshold NMOS and PMOS devices are defined in addition to the original NMOS and PMOS models. Threshold voltages of these new devices are defined as $|V_t| = 0.25$ V. Symbols for the original (high threshold) and new (low threshold) transistors are given in Fig. 2. In this study, voltage values corresponding to ternary logic levels 2, 1, and 0 are $V_{dd} = 1$ V, $V_{bb} = 0.5$ V and 0 V (ground), respectively.

![Symbols of low and high $V_t$ transistors.](image)

We implement the proposed ternary TLG in Fig. 1(a) by combining ternary inverters with a differential TLG circuit. To feed the ternary signal to TLG, we decompose ternary signals into binary form. As an example, the ternary signal $a$ can be expressed in terms of binary signals $a_1$ and $a_2$ as $a = (a_1 + a_2)/2$

where

$$a_1 = \begin{cases} 2 & \text{if } a \geq 1 \\ 0 & \text{if } a = 0 \end{cases}$$

$$a_2 = \begin{cases} 2 & \text{if } a = 2 \\ 0 & \text{if } a \leq 1 \end{cases}$$

We can obtain the decomposed binary signals $a_1$ and $a_2$ by first applying negative ternary inverter (NTI) and positive ternary inverter (PTI) and then binary inverters (INV) as in Fig. 3.2. We provide definitions of the negative ternary inverter, $a^-$, and positive ternary inverter, $a^+$, in Eqns. 5 and 6, respectively. We provide transistor level implementation of these inverters in Appendix A.1.

$$a^- = \begin{cases} 2 & \text{if } a = 0 \\ 0 & \text{if } a \neq 0 \end{cases}$$

$$a^+ = \begin{cases} 2 & \text{if } a \neq 2 \\ 0 & \text{if } a = 2 \end{cases}$$

We can obtain $b_1, b_2, c_1, c_2, d_1,$ and $d_2$ signals in the same way from ternary inputs $b, c,$ and $d$ of the TLG in Fig. 1(a). Then, we feed these signals to the differential TLG in Fig. 4. This way, we can rewrite Eqn. 1 as

$$o = \text{sgn}(n_1 \cdot (a_1 + a_2)/2 + n_2 \cdot (b_1 + b_2)/2 + \varepsilon - n_3 \cdot (c_1 + c_2)/2 - n_4 \cdot (d_1 + d_2)/2) + 1$$

(7)
The layout in Fig. 4 is a well known differential TLG circuit which works in two phases controlled by a clock [1, 2]. When the clock is high, internal nodes connected to the comparator are discharged to ground. When the clock goes low, transistors controlled by the signals $a_i$, $b_i$, $c_i$, and $d_i$ charge internal nodes. The current in-balance between left and right sides of the comparator forces one side to fall to ground. We always inject a small amount of additional current to the right side of the circuit in our implementation. This corresponds to the small positive offset $\varepsilon$ in Eqn. 7.

To sum up, we implement the proposed ternary TLG by combining the differential TLG circuit in Fig. 4 with the circuit in Fig 3.2 and its variants for $b$, $c$, and $d$. The circuit in Fig. 4 is level sensitive to clock. We append a binary SR-latch to it to make it edge sensitive. As a result, the proposed TLG based implementation comes with a bonus flip-flop.

4 Ternary Logic Gates

Ternary logic gates can be implemented utilizing the proposed ternary TLG. In this section, we introduce ternary TLG based implementations for the ternary inverter and ternary two input AND, OR, and XOR gates. M input gates can be implemented by combining these two input gates hierarchically.

4.1 Inverter

Standard ternary inverter (STI) operation on a trit $x$ is defined as $x^\sim = 2 - x$. We implement this function using the proposed ternary TLG in as Fig 5. In this
Implementation and Applications of a Ternary Threshold Logic Gate

circuit, outputs of TLGs control transmission gates. If the input $x$ is at logic level 2, the output $x^\sim$ is connected to ground which corresponds to logic level 0. When $x$ is at logic level 1, the output is connected to $V_{bb}$ which represents logic level 1. Similarly, the output is shorted to $V_{dd}$ (logic level 2), when $x$ is at logic level 0.

Fig. 5 Implementation of STI via the ternary TLG.

4.2 AND Gate

There are different definitions of the ternary AND gate. We pick the min(·) function as the ternary AND operation in this study. We can re-frame the mathematical representation for this gate as

$$\min(x, y) = \frac{x}{2}(1 + \text{sgn}(y - x + \varepsilon)) + \frac{y}{2}(1 + \text{sgn}(x - y + \varepsilon))$$

where sgn(·) is the sign function, a small offset $\varepsilon$ is added to prevent the sign function yielding 0. We benefit from the proposed ternary TLG and transmission gates to implement the ternary AND gate as in Fig. 6. Here, ternary D-latches are used to synchronize inputs $x$ and $y$ with the clock of the TLG. Implementation of the ternary D-latch is provided in Appendix A.3.

Fig. 6 AND gate (min function) implementation for ternary logic utilizing TLG.
4.3 OR Gate

There are different definitions of the ternary OR gate. We use the $\max(\cdot)$ function as the ternary OR operation in this study. We can re-formulate the mathematical representation of this gate as

$$\max(x, y) = \frac{x}{2}(1 + \text{sgn}(x - y + \varepsilon)) + \frac{y}{2}(1 + \text{sgn}(y - x + \varepsilon))$$

where $\text{sgn}(\cdot)$ is the sign function and $\varepsilon$ is a small offset. We benefit from the proposed ternary TLG and transmission gates to implement the $\max(x, y)$ function as in Fig. 7. This implementation is similar to the AND gate implementation. However, control signals of the transmission gates are flipped here.

![Fig. 7 OR gate (max function) implementation for ternary logic utilizing TLG.](image)

4.4 XOR Gate

We benefit from the definition in [25] for the XOR gate. Hence, output $s$ of the ternary XOR gate is defined as the modulo-3 sum of two trits $x$ and $y$. The generated carry is ignored. Truth table of the ternary XOR gate is provided in Table 1.

| $x$ | $y$ | $s$ |
|-----|-----|-----|
| 0   | 0   | 0   |
| 0   | 1   | 1   |
| 0   | 2   | 2   |
| 1   | 0   | 1   |
| 1   | 1   | 2   |
| 1   | 2   | 0   |
| 2   | 0   | 2   |
| 2   | 1   | 0   |
| 2   | 2   | 1   |
We implement the ternary XOR gate using the proposed TLG as in Fig. 8. As can be seen in this figure, we short circuit output to ground (logic level 0), or to $V_{bb}$ (logic level 1), or to $V_{dd}$ (logic level 2) via transmission gates. Control signals of transmission gates are generated by ternary TLGs.

![Fig. 8 Implementation of the ternary XOR gate via TLG.](image)

5 Comparing Ternary Numbers

We form a setup to compare two ternary numbers via the proposed ternary TLG. First, we start with the comparison of two trits. Then, we extend this setup to compare two ternary numbers with $M$ digits.

5.1 Comparison of Two Trits

We can compare two trits, $x$ and $y$, benefiting from the proposed ternary TLG. The circuit setup for the comparator is as in Fig. 9. In this figure, $x$ and $y$ are connected to inputs $a$ and $d$ of the TLG, respectively. If the output $o$ is at logic level 2, then we know that $x \geq y$. Otherwise, $x < y$.

If we swap inputs $x$ and $y$ in Fig. 9, the modified TLG circuit will check $y \geq x$. Thus, using the circuit in the figure and its modified version, we can simultaneously check the conditions $x \geq y$ and $y \geq x$. Moreover, the proposed TLG allows us to use binary logic gates AND ($\land$) and OR ($\lor$) since it outputs logic level 0 (false) or 2 (true). This leads to controlling $x = y$ as $(x \geq y) \land (y \geq x)$.
5.2 Comparison of Two Ternary Numbers

The comparator in Fig. 9 can be used to setup a circuit to compare two ternary numbers with M digits. In fact, this comparison is the same as the one used in binary logic. Let’s explain it by an example. Assume that we have two ternary numbers \( x \) and \( y \) with two digits each. Hence, we will have \( x = x_2x_1 \) and \( y = y_2y_1 \) where \( x_i \) and \( y_i \) for \( i = 1, 2 \) are trits. We can check whether \( x > y \) by

\[
(x_2 > y_2) \lor ((x_2 = y_2) \land (x_1 > y_1))
\]  
(10)

We can extend this operation to two ternary numbers with M digits in the same way.

6 Ternary Arithmetic Operations

We can implement ternary arithmetic operations via the proposed ternary TLG. In this section, we consider the ternary half-adder implementation first. Then, we provide the ternary complement operation to use the same circuitry in implementing the subtraction operation.

6.1 Ternary Half Adder

We propose a ternary half adder (THA) circuit with the proposed ternary TLG in this section. The reader may find truth table of the THA in Appendix A.2. We benefit from the ternary XOR gate presented in Section 4.4 to calculate the sum trit in the proposed implementation. We use the TLG circuit in Fig. 10 to calculate the carry trit.

6.2 Ternary Half Subtractor

Complement based subtraction operation used in binary logic applies to ternary (and in general Nary numbers) in the same way. For the ternary case, one’s complement in binary logic is replaced by two’s complement in ternary
logic. Two’s complement in binary logic is replaced by three’s complement in ternary logic. To perform these operations, we just need ternary inversion. Here, we will use the STI introduced in Section 4.1. Then, we will use THA for complement based subtraction. We should also increment the result. Here, we can use THA as well.

7 Comparison of Designs

We provide comparison of designs in this section. We handle this in two steps. First, we compare the TLG based ternary inverter, comparator, AND, OR, and XOR gates with their standard (CMOS only) implementations. Second, we handle the THA circuit which is frequently used to compare the performance of CNTFET ternary circuits [21]. In the same line, we compare our THA with the previously reported THA circuits in this section. We provide the simulation files for all our designs in the GitHub link https://github.com/ahmetunu/Ternary-TLG.git. Hence, the reader can compare the provided designs with theirs in the future.

7.1 Inverter, Comparator, AND, OR and XOR Gates

We compare the TLG based ternary inverter (STI), comparator (COMP), AND, OR, and XOR gates with their standard (CMOS only) implementation in terms of simulations in this section. We provide the standard implementation of STI in Appendix A.1. We use the method in Appendix A.2 to implement the standard version of COMP, AND, OR, and XOR gates.

We performed all simulations by setting the clock speed to 1 GHz. Besides, all outputs are loaded with 1 fF capacitor. The proposed ternary TLG inherently includes a flip-flop at its output. Therefore, flip-flops are appended to standard implementations to obtain the same functionality and to make a fair comparison. We provide the comparison results between the proposed ternary TLG based designs and their standard implementations in Table 2. In this table, STD+FF stands for the standard implementation with flip-flops. TLG stands for the proposed ternary TLG based implementation.

| Circuit | Type   | Static P. | Power | Delay | PDP | Area |
|---------|--------|-----------|-------|-------|-----|------|
| STI     | STD+FF | 19.11     | 10.85 | 270   | 29.30 | 256.77 |
|         | TLG    | 9.25      | 7.41  | 197   | 14.60 | 36.54 |
| COMP    | STD+FF | 4.63      | 7.51  | 255   | 19.15 | 36.72 |
|         | TLG    | 4.78      | 3.34  | 171   | 5.71  | 4.05  |
| AND     | STD+FF | 21.24     | 6.93  | 311   | 21.55 | 246.96 |
|         | TLG    | 21.25     | 9.77  | 288   | 28.13 | 238.81 |
| OR      | STD+FF | 21.24     | 7.06  | 309   | 21.82 | 240.26 |
|         | TLG    | 21.25     | 9.95  | 304   | 30.25 | 238.81 |
| XOR     | STD+FF | 19.42     | 14.72 | 292   | 42.98 | 259.38 |
|         | TLG    | 24.49     | 13.65 | 151   | 20.61 | 73.08 |
As can be seen in Table 2, the proposed ternary TLG based designs outperform standard implementations on STI, COMP, and XOR gates. Moreover, proposed ternary TLG based designs for the mentioned circuits offer significant chip area reduction as well as improvement in power consumption and delay. These results are promising and highlight the potential of the proposed ternary TLG based designs. On the other hand, delay and chip area of the TLG based AND and OR gates are similar to their standard implementations. This is mainly due to ternary latches added to these gates for synchronization. If an implementation without those latches can be done, then these gates may also outperform their standard implementation. To note here, ternary TLG based design is completely compatible with standard implementation. Therefore, the designer may use ternary TLG based circuits and standard implementations in the same design.

7.2 Half Adder

We focus on the comparison of proposed TLG based implementation of THA in this section. To the authors knowledge, this is the first study which uses a modern multi-Vt MOSFET technology to implement ternary logic. Therefore, multi-Vt MOSFET ternary designs are not available for comparison. Hence, we compare our design with the CNTFET based designs. To do so, we benefit from the results provided in [21]. We also compare TLG based design with its standard implementation given in Appendix A.2.

During comparison, all simulations are performed at 500 MHz and all outputs are loaded with 1 fF capacitor to be consistent with the setup in [21]. We provide the comparison results in Table 3. In this table, we compare power consumption and delay of the proposed ternary TLG based THA implementation with the standard implementation (STD), standard implementation having flip-flops (STD+FF), and the ones in literature [5, 16–18, 21]. Unfortunately, CNTFET based designs are abstract and their chip area is not available. Therefore, we could not perform the area comparison for the THA.

| Ternary Half Adder | Device Type | Flip Flop | Static P. \( \times 10^{-7} \) W | Power \( \mu \) W | Delay \( \times 10^{-12} \) s | FDP \( \times 10^{-16} \) J |
|-------------------|-------------|-----------|-------------------------------|----------------|-----------------|--------------|
| Bastani et al. [16] | CNTFET | No | 52.5 | 2.74 | 69.8 | 1.91 |
| Lin et al. [5] | CNTFET | No | 76.8 | 4.00 | 65.9 | 2.64 |
| Srinivasu et al. [17] | CNTFET | No | 183.6 | 95.88 | 72.2 | 69.24 |
| Sahoo et al. [18] | CNTFET | No | 16.2 | 1.15 | 50.1 | 0.57 |
| Sharma et al. [21] | CNTFET | No | 6.7 | 1.43 | 39.7 | 0.57 |
| STD MOSFET | No | 4.9 | 1.19 | 184.2 | 4.40 |
| STD+FF MOSFET | Yes | 19.5 | 8.56 | 243.6 | 20.85 |
| TLG (Proposed) MOSFET | Yes | 24.5 | 7.77 | 185.9 | 14.44 |

We can analyze the results in Table 3 in detail. Beforehand, we should mention that none of the CNTFET based designs have a flip-flop at their output. Therefore, the comparison is significantly biased and favors CNTFET
implementations. To reduce this bias, we included a standard (CMOS only) implementation (STD) without flip-flops. Comparing the standard implementation and CNTFET designs, we observe that power consumption of these are similar. Hence, we expect to see similar power figures for the ternary TLG based design and CNTFET designs if flip-flops are added to CNTFET implementations.

Comparing the delay of CNTFET based designs with the standard implementation in Table 3, we may conclude that CNTFET technology offers less delay. Although CNTFET technology seems promising, large scale CNTFET designs do not seem to be achievable in the near future. The largest CNTFET design is presented in [7] and it only contains 15,000 CNTFETs. Moreover, integrating CNTFETs with different threshold voltages is an additional challenge. On the other hand, designs implemented with recent CMOS technology nodes contain several billions of transistors. Moreover, these technology nodes readily support multi-threshold designs. Thus, in contrary to CNTFET based designs, our designs based on the proposed TLG can be realized physically. Therefore, we have a significant advantage compared to CNTFET based designs. As can be seen in Table 3, our ternary TLG based THA design decreases power consumption and delay compared to standard implementation with the same functionality (STD+FF).

8 Final Comments

In this study, we propose the first implementation of a ternary TLG circuit which is designed using a multi-Vt CMOS process. Using this gate, we implemented the ternary inverter, comparator, AND, OR, and XOR gates, arithmetic operators for ternary addition and subtraction. These modules can be taken as subparts of an ALU working on ternary representations. We compared the proposed TLG based implementations with their standard (CMOS only) versions and the ones in literature. We picked the design metrics as power consumption, delay, and chip area in comparison. The proposed TLG based inverter, comparator, XOR gates and THA outperformed their standard implementations. Moreover, TLG based design come with a bonus flip-flop. This makes it very suitable for RTL designs, where flip-flops are frequently required. However, the proposed ternary TLG based implementation could not yield a significant improvement for the ternary AND and OR gates due to additional latches. Hence, a design methodology is needed to decide when to use the proposed ternary TLG. Utilizing such a design methodology, appropriate parts of a ternary CMOS may be replaced with the TLG. This may lead to improving the delay, power consumption, and area of ternary circuits.

References

[1] Leshner, S. et al. A low power, high performance threshold logic-based standard cell multiplier in 65 nm CMOS (2010). IEEE Computer Society Annual Symposium on VLSI.
[2] Kulkarni, N., Yang, J., Seo, J.-S. & Vrudhula, S. Reducing power, leakage, and area of standard-cell asics using threshold logic flip-flops. *IEEE Transactions on Very Large Scale Integration Systems* **24** (9), 2873–2886 (2016).

[3] Saxena, P., Shelar, R. S. & Sapatnekar, S. *Routing Congestion in VLSI Circuits: Estimation and Optimization* (Springer Science & Business Media, 2007).

[4] Balla, P. C. & Antoniou, A. Low power dissipation MOS ternary logic family. *IEEE Journal of Solid-State Circuits* **19** (5), 739–749 (1984).

[5] Lin, S., Kim, Y.-B. & Lombardi, F. CNTFET-based design of ternary logic gates and arithmetic circuits. *IEEE Transactions on Nanotechnology* **10** (2), 217–225 (2009).

[6] Gadgil, S. & Vudadha, C. Design of CNTFET-based ternary ALU using 2:1 multiplexer based approach. *IEEE Transactions on Nanotechnology* **19**, 661–671 (2020).

[7] Hills, G. *et al.* Modern microprocessor built from complementary carbon nanotube transistors. *Nature* **572** (7771), 595–602 (2019).

[8] Chang, V. S. *et al.* Enabling multiple-vt device scaling for CMOS technology beyond 7nm node (2020). IEEE Symposium on VLSI Technology.

[9] Beiu, V., Quintana, J. & Avedillo, M. VLSI implementations of threshold logic—a comprehensive survey. *IEEE Transactions on Neural Networks* **14** (5), 1217–1243 (2003).

[10] Neutzling, A., Matos, J. M., Mishchenko, A., Reis, A. & Ribas, R. P. Effective logic synthesis for threshold logic circuit design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **38** (5), 926–937 (2018).

[11] Mazumder, P., Kang, S.-M. & Waser, R. Memristors: devices, models, and applications. *Proceedings of the IEEE* **100** (6), 1911–1919 (2012).

[12] Maan, A. K., Jayadevi, D. A. & James, A. P. A survey of memristive threshold logic circuits. *IEEE Transactions on Neural Networks and Learning Systems* **28** (8), 1734–1746 (2016).

[13] Yang, J., Kulkarni, N., Yu, S. & Vrudhula, S. Integration of threshold logic gates with RRAM devices for energy efficient and robust operation (2014). IEEE/ACM International Symposium on Nanoscale Architectures.
[14] Vrudhula, S., Kulkami, N. & Yang, J. Design of threshold logic gates using emerging devices (2015). IEEE International Symposium on Circuits and Systems.

[15] Papandroulidakis, G., Serb, A., Khiat, A., Merrett, G. V. & Prodromakis, T. Practical implementation of memristor-based threshold logic gates. *IEEE Transactions on Circuits and Systems I: Regular Papers* **66** (8), 3041–3051 (2019).

[16] Bastani, N. H., Moaiyeri, M. H. & Navi, K. Carbon nanotube field effect transistor switching logic for designing efficient ternary arithmetic circuits. *Journal of Nanoelectronics and Optoelectronics* **12** (2), 118–129 (2017).

[17] Srinivasu, B. & Sridharan, K. Low-complexity multiterminal digit multiplier design in CNTFET technology. *IEEE Transactions on Circuits and Systems II: Express Briefs* **63** (8), 753–757 (2016).

[18] Sahoo, S. K., Akhilesh, G., Sahoo, R. & Muglikar, M. High-performance ternary adder using CNTFET. *IEEE Transactions on Nanotechnology* **16** (3), 368–374 (2017).

[19] Soliman, N. S., Fouda, M. E. & Radwan, A. G. Memristor-CNTFET based ternary logic gates. *Microelectronics Journal* **72**, 74–85 (2018).

[20] Soliman, N. et al. Ternary functions design using memristive threshold logic. *IEEE Access* **7**, 48371–48381 (2019).

[21] Sharma, T. & Kumre, L. CNTFET-based design of ternary arithmetic modules. *Circuits, Systems, and Signal Processing* **38** (10), 4640–4666 (2019).

[22] Srinivasu, B. & Sridharan, K. A synthesis methodology for ternary logic circuits in emerging device technologies. *IEEE Transactions on Circuits and Systems I: Regular Papers* **64** (8), 2146–2159 (2017).

[23] Zhao, W. & Cao, Y. New generation of predictive technology model for sub-45 nm early design exploration. *IEEE Transactions on Electronic Devices* **53** (11), 2816–2823 (2006).

[24] Predictive technology model (ptm). URL https://ptm.asu.edu/.

[25] Murotiya, S. L. & Gupta, A. Design of high speed ternary full adder and three-input xor circuits using cntfets. *2015 28th International Conference on VLSI Design* **292–297** (2015).
Appendix A  Standard Implementation of Ternary Functions

We provide the standard (CMOS only) implementations of ternary functions in this section. Our aim here is twofold. First, some of these designs have been used in implementation of the proposed ternary TLG. Second, we compare these standard designs with the ones proposed in this study.

A.1 Ternary Inverters

Transistor level implementation of the ternary inverters NTI, PTI, and STI are given in Fig. A1. In these designs, low and high threshold devices are used. The INV gate in the figure represents the binary NOT operation.

![Transistor level implementation of ternary inverters NTI, PTI, and STI.](image)

Fig. A1  Transistor level implementation of ternary inverters NTI, PTI, and STI.

A.2 General Ternary Function

A ternary function, such as THA, can be implemented via CMOS logic. Therefore, we should check the truth table of ternary inverters and their derivatives as given in Table A1.

Table A1  Truth table of ternary inverters, NTI, STI, and PTI.

|   | x | x− | x+ | (x−)~ | (x+)~ | x+ ∧ (x−)~ |
|---|---|----|----|-------|-------|-------------|
| 0 | 2 | 2  | 2  | 0     | 0     | 0           |
| 1 | 0 | 1  | 2  | 2     | 0     | 2           |
| 2 | 0 | 0  | 0  | 2     | 2     | 0           |

As can be seen in Table A1, \(x^−\) is active iff \(x = 0\). \((x^+)\) is active iff \(x = 2\). \(x^+ ∧ (x^+)\) is active iff \(x = 1\). Therefore, these three representations are sufficient to decode \(x\) as

\[
x = \begin{cases} 
0 & \text{if } x^− \\
1 & \text{if } x^+ ∧ (x^+) \\
2 & \text{if } (x^+)\ 
\end{cases} \quad (A1)
\]
Based on the decoding scheme in Eqn. A1, a general structure to implement a ternary function can be formalized as in Fig. A2. There are three networks in this figure labeled as 1, 2, and 3. The first network is activated when $y(a) = 1$. In this case, this network pulls the output $y$ to $V_{bb}$. The second network is activated when $y(a) = 2$. In this case, the network pulls the output $y$ to $V_{dd}$. The third network is activated when $y(a) = 0$. In this case, the third network pulls the output $y$ to ground.

![Fig. A2 General structure to implement a ternary function.](image)

As an example, we can implement the THA following the presented methodology. Truth table of the THA and decoded signals are given in Table A2. In this table, $c$ and $s$ stand for the carry and sum trits, respectively.

| x | y | c | s | $x^-$ | $y^-$ | $(x^+)^\sim$ | $(y^+)^\sim$ | $x^+ \land (x^-)^\sim$ | $y^+ \land (y^-)^\sim$ |
|---|---|---|---|-------|-------|------------|------------|----------------|----------------|
| 0 | 0 | 0 | 0 | 2     | 2     | 0          | 0          | 0              | 0              |
| 0 | 1 | 0 | 1 | 2     | 0     | 0          | 0          | 0              | 2              |
| 0 | 2 | 0 | 2 | 2     | 0     | 0          | 2          | 0              | 0              |
| 1 | 0 | 0 | 1 | 0     | 2     | 0          | 0          | 2              | 0              |
| 1 | 1 | 0 | 2 | 0     | 0     | 0          | 0          | 2              | 2              |
| 1 | 2 | 1 | 0 | 0     | 0     | 0          | 2          | 2              | 0              |
| 2 | 0 | 0 | 2 | 2     | 0     | 0          | 0          | 0              | 0              |
| 2 | 1 | 1 | 0 | 0     | 2     | 0          | 0          | 2              | 2              |
| 2 | 2 | 1 | 1 | 0     | 0     | 2          | 2          | 0              | 0              |

As can be seen in Table A2, the $c$ trit is 0 when $x^- \lor y^- \lor (x^+ \land (x^-)^\sim \land y^+)$ is true. Based on this, we can implement the pull down network (network 3). Similarly, we can design the remaining networks (network 2 and 3) for the carry output $c$ of THA. The circuit to generate the carry $c$ is shown in Fig. A3. Following a similar procedure, we can implement the circuit for the sum output $s$ of the THA.

### A.3 Ternary D-Latch

Ternary D-latch may be implemented by using two cross coupled STI gates given in Appendix A.1. Circuit implementation of the ternary D-latch is as in Fig. A4. Working principles of this latch can be explained based on the circuit in this figure as follows. When the clock signal is high, the transmission gate
Fig. A3 Implementation of the carry generator circuit.

on the left side of the circuit conducts and the transmission gate on the right side turns off. Hence, the feedback loop is cut. When the clock signal is low, connection to the input is cut and the feedback is activated. Thus, two cross coupled STI gates can hold trit values 0, 1, or 2. By concatenating two ternary latches, a ternary D flip-flop can be implemented.

Fig. A4 Implementation of the ternary D-latch.