B2T: The Third Logical Value of a Bit

Dipesh
Dept. of CSE
IIT Kanpur, India
dipesh@cse.iitk.ac.in

Vishesh Mishra
Dept. of CSE
IIT Kanpur, India
vishesh@cse.iitk.ac.in

Urbi Chatterjee
Dept. of CSE
IIT Kanpur, India
urbic@cse.iitk.ac.in

***Abstract*—Modern computing systems predominantly operate on the binary number system that accepts only '0' or '1' as logical values leading to computational homogeneity. But this helps in creating leakage patterns that can be exploited by adversaries to carry out hardware and software-level attacks. Recent research has shown that ternary systems, operating on three logical values ('0, '1', and 'z') can surpass binary systems in terms of performance and security. In this paper, we first propose a novel approach that assigns logical values based on the direction of current flow within a conducting element, rather than relying on the voltage scale. Furthermore, we also present the mathematical models for each ternary gate.**

***Index Terms*—Ternary System, HDL, Mathematical Model**

I. Introduction

Modern computers operate on a binary system that consists of only two logical values, namely '0' and '1', which correspond to a true or false state. However, with moore's law reaching its limits, it has been difficult to cater to the ever-increasing computational demands through conventional binary-based computations. In addition, binary system facilitates a pattern of computational homogeneity that results in a threat to the secret key entropy in the majority of cryptographic implementations [1], [2].

Modern computer applications, particularly machine learning, artificial intelligence, and scientific computing, generate substantial data computations that strain binary-based systems. To overcome these challenges, researchers have been exploring alternative computing approaches, including quantum computing, in-memory computing, and approximate computing, to make computation faster and more secure. Moreover, they have shown increasing interest in alternative number systems, offering advantages beyond the traditional binary system. Research also shows that ternary-based cryptography performs better than traditional binary-based cryptography not only in terms of computation and energy [3] but also in terms of security [4], [5]. In response to these opportunities, recent works [6] have emerged, focusing on designing and implementing the ternary number system with a radix of three. This ternary system shows promising advantages over the binary system regarding computation and security.

Unlike previous studies, we investigate using three ternary logic values $(0, 1,$ and $z)$ within existing binary logic gates. By incorporating ternary logic into the binary framework, we aim to expand design possibilities and improve logic circuit functionality. The key contributions of this work are as follows:

- We adapt the hardware to implement ternary number systems by utilizing high impedance 'Z' in conducting elements as a third logical value of a bit.
- Unlike traditional transistor-level designs, our unique approach integrates ternary logic into existing logic gates. We validated this using Hardware Description Language (HDL) Verilog and formulated the mathematical functionalities of logical gates within the ternary system.

II. Proposed Ternary System

The existing binary system works on two logical values: 0 and 1. At the hardware level, zero corresponds to the absence of an electrical signal or a low voltage level (usually close to 0 volts), while one represents the presence of an electrical signal or a high voltage level (usually at a specified voltage, often referred to as the logic-high level). Alternatively, it can also be attributed to the current movement inside a conducting element. For instance, the current movement from anode to cathode due to the high potential difference (e.g., 5V) is considered logical '1'. If the movement is in the reverse direction (reverse saturation current) or in the same direction (but due to significantly low potential difference), it is considered logical '0'. Interestingly, there can be a third possibility as well. For example, there is a high impedance inside the wire, (represented using 'Z' in Verilog HDL) or the flowing current is significantly low (i.e., less than a predefined threshold value) which can be used as our third logical value for a bit. This way, the conducting element now has three possible values viz., 0,1, and Z depending upon the net movement of electrons inside it. The same convention can be used to define the state of storage elements e.g. capacitor, which is defined as follows:

- Logical 1; if $Q/C >= T$.
- Logical 0; if $Q/C < T$.
- X as used in HDL like verilog; if $Q = 0$.

Using symbols Q, C, T, we relate to the charge, capacitance, and threshold voltage of a capacitor. Unlike binary systems limited to 0 and 1, we introduce a third option, Z, for wires, and X for registers. This extends to a new number system where elements can be 0, 1, Z, or X. This introduces a ternary system, offering three possible values to depict the state of each conductor or storage unit. Figure 1, 2, and 3 briefly explain the movement of electrons corresponding to these logical values.

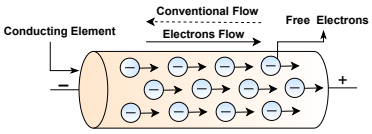

Fig. 1. Illustration of the movement of electrons inside the conducting element (from cathode to anode). This shows that the current direction is from anode to cathode, and the negative potential difference between cathode and anode is significantly high. This is considered a logical 0 in the binary representation.

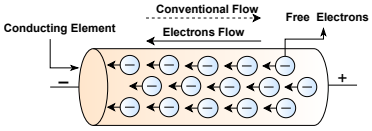

Fig. 2. Illustration of the movement of electrons inside the conducting element (from anode to cathode). This shows that the current direction is from cathode to anode, and the potential difference between cathode and anode is significantly high. This is considered logical 1 in the binary representation.

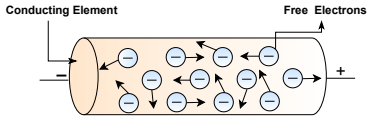

Fig. 3. Illustration of the movement of electrons inside the conducting element is entirely random, and as a net result, they cancel out each other. This means there is no net movement (or neglectable) of electrons inside the conducting element. This results in a high impedance denoted as Z in conventional hardware description language like Verilog.

A. Logical gates behaviour with ternary inputs

The binary input behavior in logical gates is well-known, our study pioneers by exploring how these gates work with ternary inputs ('Z/X').To conduct our research, we adopted a hit-and-try methodology, introducing a third logical value alongside the conventional binary values and meticulously observing the resulting outputs for each logical gate. Our aim is to grasp the influence of ternary inputs on the overall performance and functionality of these gates during ASIC synthesis. For the simulations, we employed advanced Computer-Aided Design tools (CAD) like Iverilog V11 to define the behaviour of circuits accurately. Using the Hardware Description Language (HDL) to assign Z or X values as inputs allows a deep examination of how existing logical gates behave with ternary inputs. Our study utilized Verilog to analyze existing logical gates like AND, OR, NOT, NOR, NAND, XOR, and XNOR.

Our research seeks to answer questions such as:

1. How does the introduction of ternary inputs impact the truth tables and logical behavior of each gate? 2. Can ternary inputs enhance the fault tolerance and reliability of digital circuits? 3. Are there any unique applications for ternary inputs in specialized computing environments? 4. What are the limitations and challenges associated with integrating ternary inputs in practical circuits?

By undertaking this innovative investigation, we hope to pave the way for the adoption of ternary inputs in logic design, opening up new possibilities for digital circuitry and potentially leading to advancements in computing systems and their applications.

In conclusion, our study aims to shed light on the functionality and viability of ternary inputs in logical gates, bringing attention to this unexplored area of research and inspiring further exploration into the world of ternary logic. The results of our experiments will contribute to the broader understanding of logic design and may spark new paradigms in digital computing.

TABLE I
TRUTH TABLE FOR LOGIC GATES WITH TERNARY INPUTS

A	B	AND	OR	NOT	$NAND$	NOR	XOR	$XNOR$
0	0	0	0	1	1	1	0	1
0	1	0	1	1	1	0	1	0
0	Z	0	X	1	1	X	X	X
0	X	0	X	1	1	X	X	X
1	0	0	1	0	1	0	1	0
1	1	1	1	0	0	0	0	1
1	Z	Z	1	0	X	0	X	X
1	X	X	1	0	X	0	X	X
Z	0	0	X	X	1	X	X	X
Z	1	Z	1	X	X	0	X	X
Z	Z	Z	X	X	X	X	X	X
Z	X	X	X	X	X	X	X	X
X	0	0	X	X	1	X	X	X
X	1	X	1	X	X	0	X	X
X	Z	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X

The results in Table I are crucial, forming a foundation to examine established logical gates with a new input set: logical 0, logical 1, and logical Z/X. While we understand outcomes for logical 1 and 0, our focus is on unraveling how gates behave when one or both inputs are logical Z/X. This helps us understand gate responses to uncertain inputs represented by logical Z/X. The experimental results are presented in Table I. In this table, columns A and B represent the inputs provided to the aforementioned logical gates. The remaining columns represent the corresponding outputs of these gates, based on the given inputs in columns A and B. It is important to note that the output of the NOT gate corresponds to the input 'A,' as it is a single-input gate.

More precisely, the findings of the experiment are as follows:

- Gates show normal behaviour with the Z/X as an input to the logical gates.
- X is shown in the output even though we have not provided X as a single input.
- Z does not appear at a single output, even if both inputs of a gate have provided Z as an input.
- No error is seen in the Verilog simulator.

Based on these findings, we draw several conclusions that are listed as follows:

- The existing logical gates are designed to work on a ternary number system.
- Both the binary and ternary number systems, can run on the same hardware.
- The register cannot hold Z as a value.
- Z and X are equivalent on the hardware level as when Z is passed to a buffer, it is converted to X, but X remains X on the same experiment.

Additional note: X is used to represent both X/Z at the same time, as they are equivalent in the findings.

Our experiments have showcased the behavior of each logic gate. Based on the behavior and its analysis, we now aim to realize each logic gate in a mathematical form with the help of a function f. Clearly, the domain and range of this function f is defined over set $B = \{-1, 0, 1\}$. Here, each element represents the net movement of electrons inside the conducting element viz., -1 represents logical 0, 1 represents logical 1, and 0 represents high impedance. The function f has the number of parameters depends upon the type of logical gate, which is 2 in most gates, except NOT gate, in which the number of parameters is 1. However, the function always returns a single value.

B. Functionality of logical gates

The functionality of each logical gate is defined using the function f. The domain and range of each function is the set B as defined above. The functionality of each logic gate is defined using the observations and findings of table I.

- **AND Gate:-** The outputs corresponding to each input are shown in Table I. As shown, '0' is returned when one input is '0', 'X' is returned when one or both of input/s is/are 'X', and another input is not '0', and '1' is returned when both inputs are '1'. From the above observation, we conclude that the maximum of the passed parameter values is returned. Consequently, functionality f is defined as:

$$f(A,B) = \begin{Bmatrix} B \ if \ A >= B \\ A \ if \ B > A \end{Bmatrix} \tag{1}$$

- **OR Gate:-** The outputs corresponding to each input are shown in Table I. As shown, '1' is returned when one input is '1', 'X' is returned when one or both of input/s is/are 'X', and another input is not '1', and '0' is returned when both inputs are '0'. From the above observation, we define the functionality f as:

$$f(A,B) = \begin{Bmatrix} A \ if \ A >= B \\ B \ if \ B > A \end{Bmatrix} \tag{2}$$

- **NOT Gate:-** The outputs corresponding to each input are shown in Table I. As shown, '1' is returned when input is '0', 'X' is returned when input is 'X', and '1' is returned when input is '0'. From the above observation, conclude that the additive inverse of the passed parameter value is returned. Also, functionality f is defined as:

$$f(A) = -A \tag{3}$$

- **NAND Gate:-** The outputs corresponding to each input are shown in Table I. As shown, '1' is returned when one input is '0', 'X' is returned when one or both of input/s is/are 'X', and another input is not '0', and '0' is returned when both inputs are '1'. From the above observation, we conclude that the additive inverse of the maximum of the passed parameter values is returned. Consequently, functionality f is defined as:

$$\begin{aligned} f(A,B) = \ & -C \\ where \ C = & \begin{Bmatrix} B \ if \ A >= B \\ A \ if \ B > A \end{Bmatrix} \end{aligned} \tag{4}$$

- **NOR Gate:-** The outputs corresponding to each input are shown in Table I. As shown, '0' is returned when one input is '1', 'X' is returned when one or both of input/s is/are 'X', and another input is not '1', and '1' is returned when both inputs are '0'. From the above observation, we are able to conclude that the additive inverse of the minimum of the passed parameter values is returned. Consequently, functionality f is defined as:

$$\begin{aligned} f(A,B) = \ & -C \\ where \ C = & \begin{Bmatrix} A \ if \ A >= B \\ B \ if \ B > A \end{Bmatrix} \end{aligned} \tag{5}$$

- **XOR Gate:-** The outputs corresponding to each input are shown in Table I. As shown, '0' is returned when both inputs are the same and not 'X', 'X' is returned when one input is 'X', and '1' is returned when both inputs are not the same and not 'X'. From the above observation, we conclude that the additive inverse of the product of the passed parameter values is returned. Also, functionality f is defined as:

$$f(A,B) = \ -(A.B) \tag{6}$$

- **XNOR Gate:-** The outputs corresponding to each input are shown in Table I. As shown, '1' is returned when both inputs are the same and not 'X', 'X' is returned when one input is 'X', and '0' is returned when both inputs are not the same and not 'X'. From the above observation, we deduce that the product of the passed parameter values is returned with functionality, f defined as:

$$f(A,B) = \ (A.B) \tag{7}$$

III. CONCLUSION

This paper highlights the practicality of employing the ternary number system on existing hardware. By analyzing the behavior of logical gates in the context of a ternary number system, we establish a mathematical model applicable to both binary and ternary systems.

REFERENCES

[1] S. Mangard *et al.*, "Successfully attacking masked aes hardware implementations," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2005, pp. 157–171.

[2] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," in *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication: 5th IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings 5*. Springer, 2011, pp. 224–233.

[3] W. Alexander, "The ternary computer," *Electronics and Power*, vol. 10, no. 2, pp. 36–39, 1964.

[4] R. Bricout *et al.*, "Ternary syndrome decoding with large weight," in *Selected Areas in Cryptography–SAC 2019: 26th International Conference, Waterloo, ON, Canada, August 12–16, 2019, Revised Selected Papers 26*. Springer, 2020, pp. 437–466.

[5] J. Adikari *et al.*, "Hybrid binary-ternary number system for elliptic curve cryptosystems," *IEEE transactions on computers*, vol. 60, 2010.

[6] S. Lin, Y.-B. Kim, and F. Lombardi, "A novel cntfet-based ternary logic gate design," in *2009 52nd IEEE International Midwest Symposium on Circuits and Systems*. IEEE, 2009, pp. 435–438.