SSRG International Journal of Electrial and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I3P109

Volume 11 Issue 3, 113-135, March 2024
© 2024 Seventh Sense Research Group®

Original Article

Low-Power VLSI Architecture for Ternary Galois Field

B.V. Srividya[1], Nagarathna[2], A.R. Aswatha[3]

[1, 2]*Department of Electronics and Telecommunication Engineering, Dayananda Sagar College of Engineering, Karnataka, India.*
[3]*Department of Medical Electronics, Dayananda Sagar College of Engineering, Karnataka, India.*

[1]*Corresponding Author : srividyabv@gmail.com*

Received: 07 January 2024 Revised: 05 February 2024 Accepted: 05 March 2024 Published: 25 March 2024

Abstract *- The cost of sending and receiving data has increased. Hence, storing and keeping the information secured has become more complex. This requires a secure connection in every application. Cryptography is one such possible solution that secures the data. In the modern world, cryptography is helpful in numerous applications such as social networking, Email, and e-commerce, in addition to defence-related ones. In predesigned interfaces like embedded systems, cryptography is the critical component. Most of the cryptographic algorithms, such as Advanced Encryption Standard (AES) and Elliptic Curve Cryptography (ECC), use modular operations involving Galois Field (GF) (p^m), where 'p', the base of the modular operations, is a prime number and 'm' represents the power. This research focuses on implementing the Ternary Galois Field (GF) (3^m) using the Forced Stack technique with low leakage power. Ternary Galois field implementation using the proposed Ternary Multiplexers and Ternary gates using the canonical expression is compared. This comparative analysis for different designs helps choose the best-optimised implementation methods concerning the number of ternary logic gates, transistors, and leakage power. Further, using the implemented Ternary Galois field elements, a module adder circuit is designed and implemented using a heuristic approach in Cadence Virtuoso.*

Keywords *- Ternary Galois Field, Low power, VLSI circuits, Galois Field adder, Cryptography.*

1. Introduction

Cryptography is the most common and extensively utilised application of Galois Field. Mathematical arithmetic makes encryption and decryption exceedingly simple and manipulable since each data byte is represented as a vector in a finite field. Data can be efficiently and readily scrambled using mathematical operations when expressed as a vector in a Galois Field.

1.1. Galois Field

Galois fields, or Finite Fields, are an abstract algebraic concept dealing with mathematics' finite structures. It is a collection of integers with a limited number of components, and two operations, such as addition and multiplication, can be performed on them, which adhere to predetermined rules. Since the Galois field is kept closed by the requirements for these operations, every operation carried out on these elements will produce a result that belongs to the same set.

Due to their unique mathematical characteristics, Galois Fields are valuable in several areas, including cryptography, coding theory, and error correction. GF (p^m), where p is a prime number, taking values such as 2, 3, 5, 7… and 'm' represents the power, indicates the size of a Galois Field. Galois field satisfies the following properties:

- Finite-size
- Closure
- Commutative
- Associative
- Distributive
- Identity elements
- Inverse elements

Hence, using Galois field elements is more appropriate for modular arithmetic computations.

1.1.1. Binary Galois Field

In Binary Galois Field (GF) (2^m), the Galois field elements are represented in Binary as a combination of two numbers 0 and 1. Let us consider GF (2^3), which has 2^3 elements $0,1,\alpha,\alpha^2, \ldots\ldots\alpha^6$. These elements are generated using a primitive polynomial $p(x) = x^3+x+1$.

By considering α as a root of the Primitive polynomial, the polynomial is evaluated as,

$$P(\alpha) = \alpha^3+\alpha+1 = 0$$
$$\text{Hence, } \alpha^3 = -\alpha-1$$

As per binary modular arithmetic,

$\alpha^3 = \alpha+1$ (1)

The elements of GF (2^3), its polynomial representation, and its Binary vector representation are indicated in Table 1. This research focuses on the Ternary Galois field and the modular addition operation.

1.1.2. Ternary Galois Field

In Ternary Galois Field (GF) (3^m), the Galois field elements are represented in Ternary as a combination of three numbers: 0, 1 and 2.

As an illustration, let us consider the Ternary Galois Field (GF) (3^2). There are nine elements in GF (3^2), as shown in Table 1. Each component is represented in the polynomial, Ternary, and decimal forms. The elements of GF (3^2) are constructed using the primitive polynomial $p(x) = x^2+x+2$. Let us assume α it to be a root of the polynomial p(x).

Hence $p(x=\alpha) = \alpha^2+\alpha+2 = 0$, $\alpha^2 = -\alpha-2$

Considering the modular operation, the element:

$\alpha^2 = 2\alpha+1$ (2)

This research aims to implement the Ternary Galois field using CMOS VLSI architecture by using Ternary Logic and implementing Ternary logic gates, which is elaborated subsequently.

Table 1. Elements of Galois Field (GF) (2^3)

Elements	Polynomial Representation	Vector Form	Decimal Representation
		The Binary Representation of the Vector from the Polynomial = $\alpha^0\alpha^1\alpha^2$	**Weightage =$2^0 2^1 2^2$**
0	0	000	(0)
1	1	100	(1)
α	α	010	(2)
α^2	α^2	001	(4)
$\alpha^3=\alpha^2*\alpha$	1+α (from Equation 1)	110	(3)
$\alpha^4=\alpha^3*\alpha$	$\alpha+\alpha^2$	011	(6)
$\alpha^5=\alpha^4*\alpha$	$1+\alpha+\alpha^2$	111	(7)
$\alpha^6=\alpha^5*\alpha$	$1+\alpha^2$	101	(5)

Table 2. The elements of GF(3^2)

Elements	Polynomial Representation	Vector Form	Decimal Representation
		Ternary Representation of the Vector from the Polynomial Weightage = $\alpha^0\alpha^1$	**Weightage = $3^0 3^1$**
0	0	$(00)_3$	(0)
1	1	$(10)_3$	(1)
α	α	$(01)_3$	(3)
α^2	1+2α	$(12)_3$	(7)
$\alpha^3=\alpha^2*\alpha$	2+2α	$(22)_3$	(8)
$\alpha^4=\alpha^3*\alpha$	2	$(20)_3$	(2)
$\alpha^5=\alpha^4*\alpha$	2α	$(02)_3$	(6)
$\alpha^6=\alpha^5*\alpha$	2+α	$(21)_3$	(5)
$\alpha^7=\alpha^6*\alpha$	1+α	$(11)_3$	(4)

1.2. Ternary Logic

Ternary logic has three levels, unlike the conventional binary logic, which has two levels. The proposed technique employs unbalanced Ternary digits, denoted as 0, 1, and 2. Compared to binary data, ternary data could fit more volumes of data in the same available storage. If binary data is represented as N bits, Ternary data is defined as N trits. For example, if N=12, there are 4096 bits, while there are 531441 trits. The primary benefit of Ternary over binary is that each wire may carry more multi-valued logic information while simultaneously taking up less space on the chip. It is given more significant consideration than binary due to the lower estimated connection cost. Discussing the various Ternary Logic gates built using CMOS techniques to implement the Ternary Galois field is essential.

Ternary Logic circuits have three different logic levels. They are logic 2, logic 1 and logic 0. In all the Ternary logic circuits discussed in this paper, logic 2 is taken as 1.8V, logic 1 is 0.9V and logic 0 is 0V during implementation.

1.3. Ternary Logic Gates Used for Implementing Ternary Galois Field

The fundamental components of all digital circuits, including multiplexers, encoders/decoders, arithmetic logic units, memory, microprocessors, and computers, are logic gates, such as AND, OR, inverters, NAND, NOR, and XOR. The development of technology has made it possible to create ternary logic circuits. Various methods, including CMOS logic, memristor logic and Carbon Nanotube Field Effect Transistors (CNTFET), are used to build ternary logic gates and digital circuits. This research work attempts to build Ternary logic gates utilising CMOS VLSI architecture.

1.4. Low Leakage Power CMOS Designs

Leakage power was disregarded in the past since it was less than dynamic power. One of the leading causes of cumulative IC currents is leakage current, which has grown significantly with the scalability of technology and cannot be ignored. Static power loss happens when the apparatus is in the OFF state or when no operations are being performed.

Providing Very Large-Scale Integration (VLSI) designers with more effective low-power solutions is the primary goal of this research. The focus is on CMOS circuit leakage power reduction techniques. At 0.18µm technology, leakage power is nearly comparable to dynamic power consumption while being low. After analysing the available leakage minimisation strategies, the network's optimal approach is implemented at the circuit level for each Ternary logic gate and logic circuit.

This article focuses on implementing the proposed low-power Ternary Galois field using Ternary gates based on the forced stack technique using GPDK045 technology in Cadence Virtuoso. The work extends to designing and implementing a Ternary Galois Field adder circuit.

The succeeding section elaborates on the survey to choose the optimised Ternary logic gate design and the best possible leakage power reduction techniques that help achieve this research's objective.

2. Literature Survey

2.1. Survey on Ternary Logic

There are various number systems to represent numbers. Humans use a decimal number system comprising digits 0 to 9 for representation. Binary has two digits, 0 and 1, representing any value. Binary representation is used widely to design and develop digital circuits, wherein logic 0 and 1 are represented by two different voltage levels [1]. Against the traditional binary system, Multiple-Valued Logic (MVL), which uses more than two symbols for the design of digital circuits, can be considered [2, 3].

Ternary and quaternary are the most widely used MVLs for digital system design-ternary and quaternary use 3 and 4 logic levels or symbols to represent a value [3]. Ternary digits 0,1,2 are called traits, and quaternary ternary digital circuits are considered to be the best choice for cost-effective and less complex systems as the number of levels (3) is close to the e (Euler's constant) [6].

Ternary representation reduces the number of digits by 36.90% compared to binary representation [4, 5]. Unbalanced Ternary represents unsigned ternary numbers, and balanced Ternary describes signed numbers. Balanced Ternary consists of -1, 0, and 1 levels to represent a value. The implementation of these ternary logic started two decades ago. SETUN- is the world's first ternary computer. Subsequently, TERNEC- a ternary computer, was implemented through program simulation on a microcomputer [7].

Logic gates are the building blocks of digital circuits like multiplexers, encoders/decoders, Arithmetic Logic Units, memories, microprocessors and computers. With technological advancements, implementations of the ternary logic circuits are achievable. Ternary logic gates and digital circuits are implemented using various techniques, namely CMOS logic, Memristors logic, Carbon Nano Tubes Field Effect Transistors (CNTFET) and many more.

2.2. Survey on Implementation of Ternary Gates

A P. Dhande et al. have implemented T-gates, i.e. TNOT, TNAND, TNOR, using a novel injected voltage technique. This work uses three different voltage levels for the representation of logic '0' (0V), logic '1' (2.5 V) and logic '2' (5V). Circuits for the gates are designed using CMOS logic with an auxiliary power supply of 2.5V connected to the output node through a resistor to pull the output to logic 1. MOSFETs' R and W/L ratios are varied to get the desired voltage levels at the output. The simulation and analysis of rise time, fall time and power consumption are carried out using the Tanner tool [6].

NAND and NOR gates are considered universal gates to build any digital circuit. V T Gaikwad et al. have designed and implemented Ternary NOT gates and universal ternary gates using CMOS logic with a single power supply. This design uses Vc - Communication voltage to decide the switching of voltage in inverters. Vc is given by the Equation 3. By varying the W/L ratios of the transistors, the expected switching voltage for the inverter is achieved.

$$\text{Vc} = \frac{k(Vtn)+VDD-Vtp}{1+k} \tag{3}$$

$$\text{Where } k = \frac{\mu n\frac{Wn}{Ln}}{\mu p\frac{Wp}{Lp}}$$

Logic gates are designed, and power and delay analysis are carried out using the Micro wind EDA tool for 45 nm technology. Using a single power supply reduces the power consumption and transistor count [8].

Ahmet Unutulmaz et al. used Threshold Logic (TLG) to implement ternary gates, comparators, and half and full adders. It uses the Multi Vt CMOS transistors technique using two additional MOSFETs called low threshold and high threshold p type and n type MOSFETS to reduce delay and power consumption. These designs with TLG are compared with CMOS ternary gates for different design metrics like chip area, delay and power consumption [9].

Ternary and Ex-OR gates are designed and synthesised using Mentor Graphics HEP-2 for 130nm technology CMOS and CNTFETs [7, 10]. Logic gates in Ternary are successfully designed and fabricated using ambipolar BP (Black Phosphorous) transistors and MoS2 transistors. These transistors are manufactured using 2D Materials (TDM) because of their unique electronic properties [11].

Half subtractor and ternary gates are designed using FinFET and simulated in the Cadence Virtuoso tool to verify operations. These gates provide fast switching speed, lower power consumption, and lower switching voltage than binary gates.

Ternary 18nm technology FinFET-based half adders are compared with its counter MOSFETs for 32nm technology, showing lesser power consumption in FinFET-based designs. Graphene-based Field Effect Transistors (GNRFETs) are used to design low-voltage logic gates, which provide robustness against variations in supply voltage and output load. The performance parameters are analysed using HSPICE [12].

Ternary gates are designed using Short Gate FinFET(SG-FinFET) using Tanner tool version 13.02. All gates' Power Delay Products (PDP) are compared for various capacitive loads. The number of transistor counts in different ternary gates is reached between CNTFET, Quantum Dot Gate FET (QDGFET) and SG-FinFETs-based designs [13, 18]. Another design proposes implementing the same using multi-threshold CNTFETs, wherein voltage division is achieved using a p-type transistor with a dynamic diode load and an n-type transistor with a static diode load. This reduces leakage current [14].

Gowri Sankar explores the implementation of gates using 32 nm Graphene Nanoribbon FET GNRFET technology and double gate FinFET transistors. Designed circuits are analysed for various supply voltages, capacitive loads and operating frequencies using HSPICE simulations [15].

Badugu Divya Madhuri et al. present the implementation of gates, half adder and full adder circuits using GNRFET transistors operating at multiple threshold values. The transistor's Vt value can be altered by controlling the carbon nanoribbon's width. These designs show reduced power dissipation, propagation delay and chip area reduction. Additional circuits like 3:1 MUX ternary decoder are also implemented using GNRFET transistors, and a comparative analysis is made between the circuits implemented using CMOS and GNRFETs, showing the reduction in PDP in the case of GNRFETs [17]

S Kim et al. have proposed some optimising techniques in implementing ternary gates. Adders and multiplier circuits use CMOS, CNTFET and QDGFET technology using proposed optimised techniques showing the reduction in transistor counts in all three [19].

2.3. Survey on Various Leakage Power Reduction Techniques

Researchers have presented multiple solutions for minimising leakage power at the architectural, circuitry, and device levels.

2.3.1. Device Level

Scaling the depth of the junction (W), the channel's length (L), and its oxide thickness (Tox) reduces the standby or leakage current while taking the device level into account [20-22]. This needs to be completed before a transistor is fabricated. Innovation in this field of study led to the development of innovative semiconductor geometries with two or more gates, such as the Fin-shaped FET (FINFET), which minimises sub-threshold power dissipation and short channel effects (I2).

2.3.2. Circuit Level

Circuit level introduction of various leakage minimisation techniques includes the following: Selective input vector technique (Input Vector Control (IVC), INDEP, bootstrapped circuit technique), mixed threshold technique (Dual Threshold MOS, Dynamic MOS, Variable Threshold MOS, Multiple Threshold CMOS, Power Gating), stacking techniques (including forced stacking, sleep stacking, LECTOR (Leakage

Controlled Transistors), GALOER, sleepy-keeper approach, and zigzag keeper approach and drain gating), biasing the source, substrate biasing, and many more.

2.3.3. Architectural Level

Techniques at the architectural level include multiple modes management, which lowers leakage current by allowing only a tiny portion of the memory (SRAM) to remain ON and putting unused memory in sleep or standby state [23, 24].

The forced stack technique gives a very low static power compared to other leakage power reduction strategies. The forced stack technique is based on the idea that two off-state transistors connected in series produce substantially less leakage than one device. The leakage current of the stack is shallow when compared to the leakage of a single device. But this expands the area. To substantiate the statement about the low static power dissipation using the forced stack technique, a conventional binary inverter and forced stack binary inverter are implemented and are verified by comparing their static power dissipation.

2.4. Conventional Binary Inverter

Truly, the inverter forms the basis of all digital designs. Extending the conclusions found for inverters makes it possible to infer the electrical behaviour of complex circuits nearly entirely. A static CMOS inverter's circuit diagram and its results are shown in Figures 1 and 2, respectively. The NMOS transistor is ON, while the PMOS transistor is OFF when Vin is high and equal to VDD. Since Vout and the ground node are connected directly, 0 V is the steady-state value.

On the other hand, NMOS and PMOS transistors are OFF and ON, respectively, when the input voltage is low (0 V). There is a path between VDD and Vout that results in a high output voltage. The gate serves as an inverter.

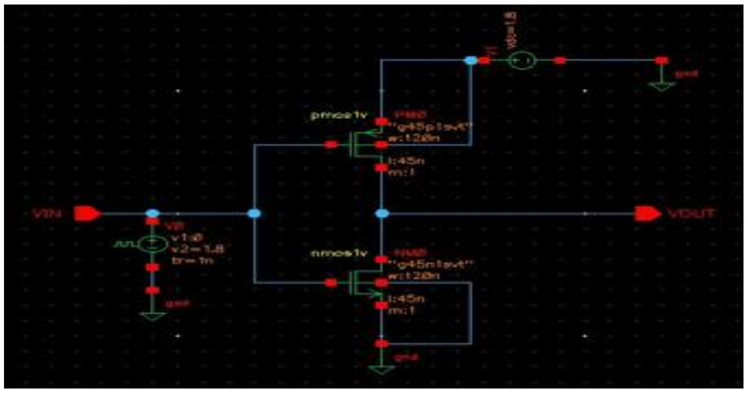


Fig. 1 Conventional inverter

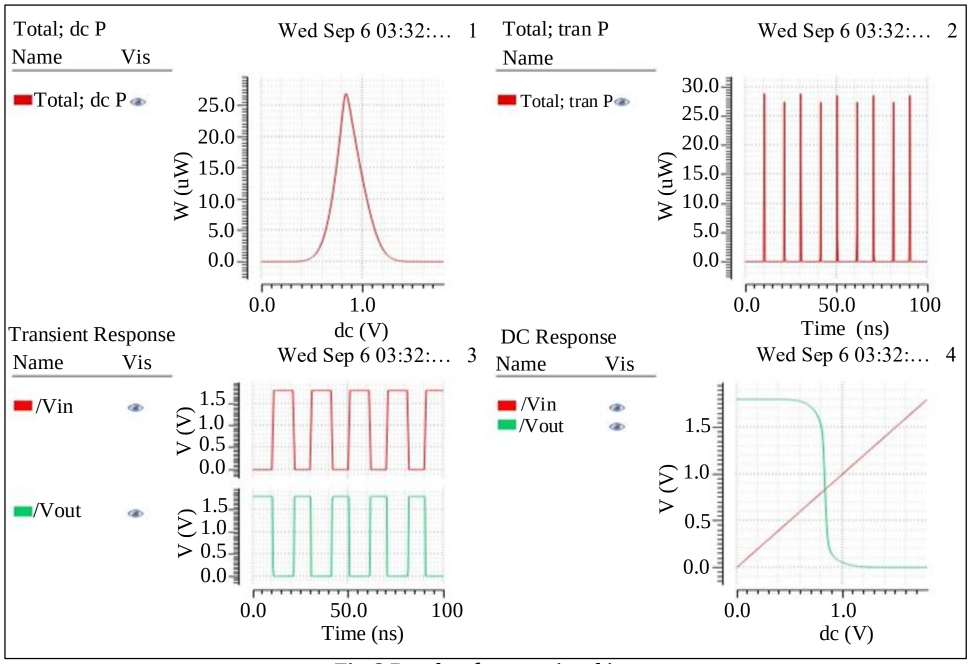


Fig. 2 Results of conventional inverter

Since low-power design has become increasingly significant in recent years and due to the rapid growth of battery-operated products, the desire for more functionality, smaller size, and longer battery life increases as electronics are built into portable gadgets. The forced Stack approach is one such technique for reducing the leakage power.

2.5. Forced Stack Binary Inverter

The forced stack method is based on the idea that two off-state transistors connected in series have significantly less leakage than a single device. The leakage current of the stack is relatively little compared to the leakage of a single device. But the delay and area grow as a result. High Vth sleep trans-resistors cross the Pull-Up and Pull-Down circuits using the sleepy transistor approach.

These sleep trans-resistors are turned ON while the electronic device is in standby mode, significantly minimising leakage current. The forced Stack Inverter has a pull-up and pull-down network of identical width, as shown in Figure 3. Both transistors are turned OFF simultaneously during the OFF state, and leakage power is minimised due to the reverse gate-to-source potential and the effect of stacking. The results of the forced stack binary inverter are shown in Figure 4. The power dissipation of the conventional binary Inverter and the forced stack binary inverter are tabulated in Table 3.

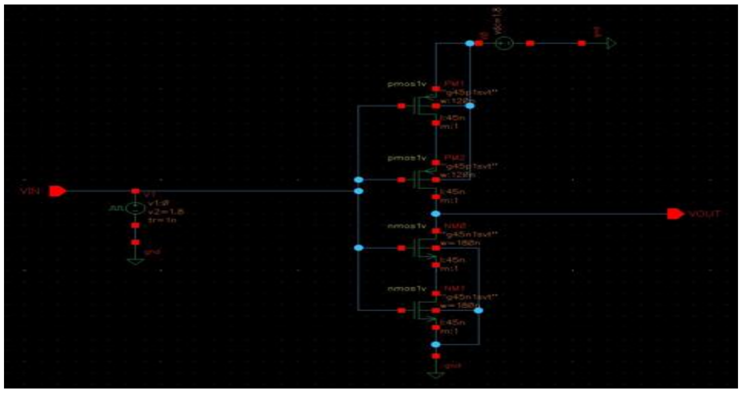


Fig. 3 Forced stack inverter

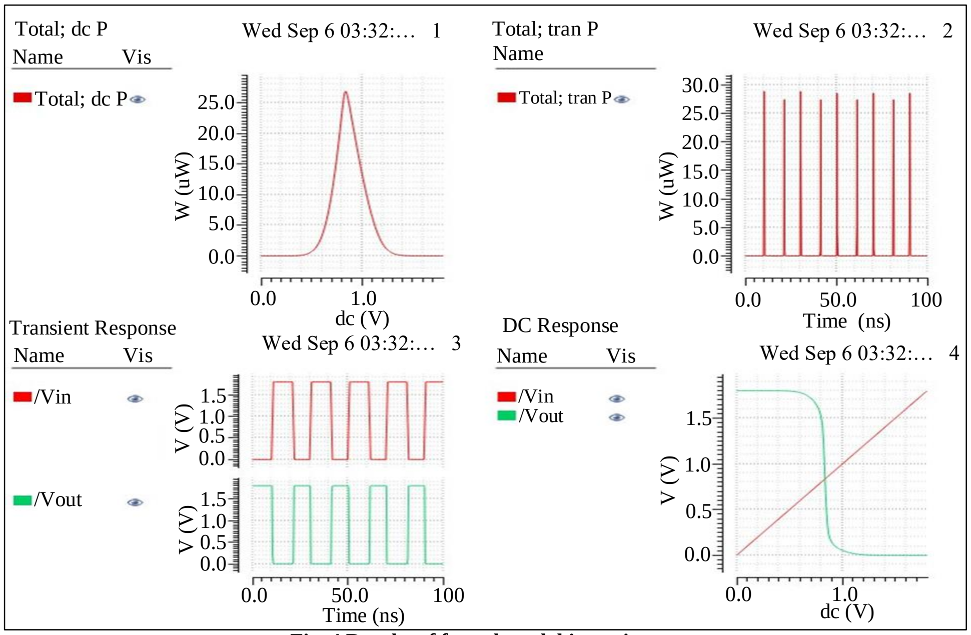


Fig. 4 Results of forced stack binary inverter

Table 3. Power analysis of binary inverters

Type of Inverter	Leakage Power	DC Power	Transient Power
Conventional Binary Inverter	6.42988 pW	50.4267 μW	51.2529 μW
Forced Stack Binary inverter	4.05841 pW	26.8913 μW	28.8536 μW

From the results obtained as per Table 3, it can be observed that the static leakage power of a forced stack binary inverter is 63.62% of that of a conventional binary inverter.

The survey shows that representing data in Ternary could fit more volumes of data in the given storage and offer enhanced data security. Moreover, when connection overhead and related latency are considered, ternary systems typically perform better than binary counterparts, especially in sophisticated data converters and processors [25]. Further, the use of the multi-threshold technique along with the forced stack technique results in low leakage power dissipation. This research uses GPDK045 technology in Cadence Virtuoso to construct the proposed low-power Ternary Galois field through the proposed low-power Ternary logic gates. The design and implementation of a ternary Galois Field Adder Circuit is another extension of the work.

3. Implementing Various Ternary Logic Gates and Ternary Circuits

3.1. Ternary Gates

To begin with, the proposed forced stack technique-based low-power Ternary logic gates with multi-threshold transistors are designed and implemented. The following section elaborates on implementing these Ternary Inverters, as they are of utmost importance due to their utilisation in most logic circuits. These multi-threshold transistors are classified as high-threshold MOSFETs, whose threshold voltage is 0.9V and low-threshold MOSFETs, whose threshold voltage is 0.4V. As an essential requirement, the proposed forced stack multi-threshold MOS-based Ternary inverter is designed and implemented as shown in Figure 5.

3.1.1. Ternary Inverter

This paper focuses on implementing the Ternary inverter field using three different designs, which includes:

a) Ternary inverter with multi threshold transistors.
b) Ternary inverter using forced stack technique and multi-threshold transistors.
c) Ternary Inverter using resistive Load.

The abovementioned cases' static power dissipation, DC power, and transient power are analysed.

Case 1: Ternary Inverter with Multi Threshold Transistors

The ternary inverter using multi-threshold transistors is shown in Figure 5. To optimise power, MTCMOS, a version of CMOS chip technology, transistors with multiple threshold voltages (Vth) are used. Low voltage threshold devices switch more quickly and may be advantageous on crucial delay paths. These devices could have significantly higher static leakage power.

On non-critical channels, high-voltage threshold devices reduce static leakage power without delay. The Ternary inverter, shown in Figure 5, uses one high threshold pMOS and nMOS and one low threshold pMOS/nMOS. When the input Vin=0, both the pMOS and the nMOS are turned ON and OFF. The output Vout is pulled to Vdc=1.8V, representing Ternary value 2.

When the input is 0.9V (representing Ternary 1), the Low threshold nMOS turns ON. The resultant circuit behaves as a voltage divider and the output Vout is pulled up to Vdc/2=0.9V, representing logic 1. When the input Vin=1.8V, the high voltage nMOS turns ON. The output is pulled down to Ground, thereby yielding Vout =0V.

The results obtained after implementation, as shown in Figure 6, follow the truth table of the Ternary inverter shown in Table 4. Figure 6 shows the simulation results obtained after implementing the Ternary inverter. The Truth table of the Ternary inverter is shown in Table 4.

Table 4. Truth table of ternary inverter

Input Vin and its Voltage Levels	Output Voltage Levels (Vout)	Ternary Output
0=(0V)	1.8V	2
1=(0.9V)	0.9V	1
2=(1.8V)	0V	0

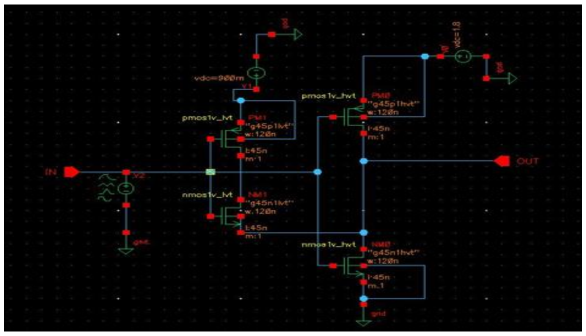


Fig. 5 Ternary inverter with multi threshold transistor

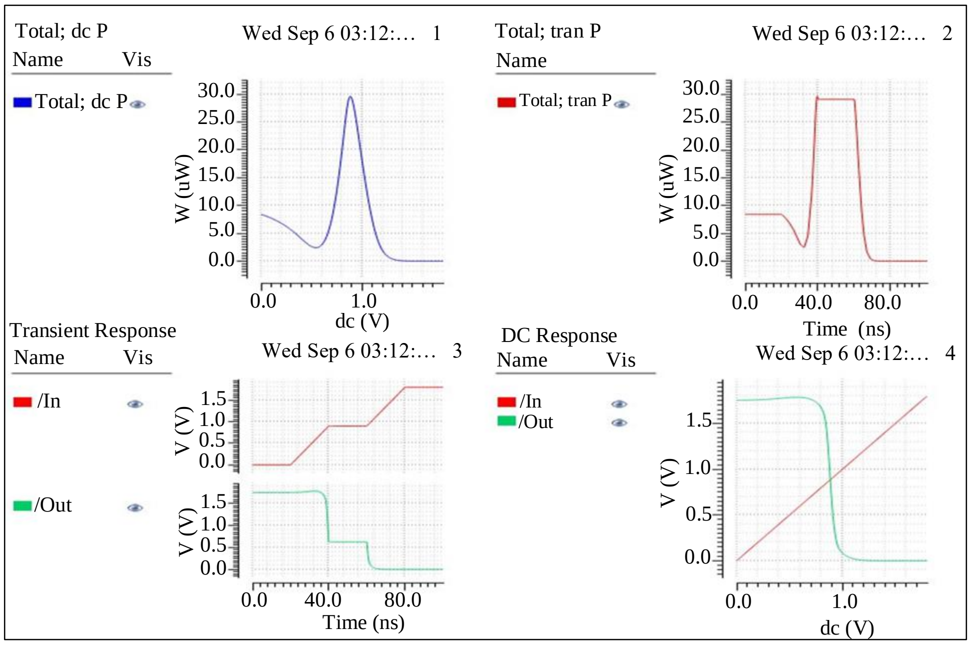


Fig. 6 Results of ternary inverter which has multi threshold transistors

Case 2: Ternary Inverter Using Forced Stack Technique and Multi-Threshold Transistors

Forced stack technique with a network of sleep transistors is the most popular method of implementing multi-threshold CMOS to save power. High Vth transistors are used to cut down on static leakage power. The proposed low-power Ternary inverter using a Forced stack approach with a threshold transistor is shown in Figure 7.

Figure 7 shows that the power Ternary inverter uses a pair of high threshold voltage pMOS and nMOS and a pair of low threshold voltage pMOS and nMOS. The forced stack inverter has a pull-up and pull-down network of identical width. High Vth sleep trans-resistors cross the Pull-Up and Pull-Down circuits using the sleepy transistor approach.

These sleep trans-resistors are turned ON while the electronic device is in standby mode, significantly minimising the leakage current. Both transistors are turned OFF simultaneously during the OFF state, and leakage power is underestimated due to the reverse gate to source potential and the effect of stacking.

By applying the input of 0V, 0.9V and 1.8V, the output obtained is 1.8V, 0.9V and 0V, respectively, as per the Truth table in Table 4. The simulation results obtained are shown in Figure 8. Implementing the Forced stack, the multi-threshold inverter gives a leakage power of 582.125fW. Hence, the forced stack multi-threshold inverter is used as a low-power Ternary inverter in the design of Ternary AND gates, OR gates and logic circuits.

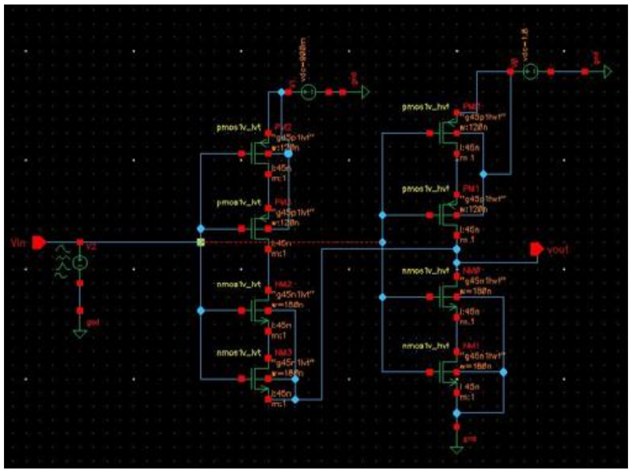


Fig. 7 Forced stack ternary inverter with multi-threshold MOSFETs

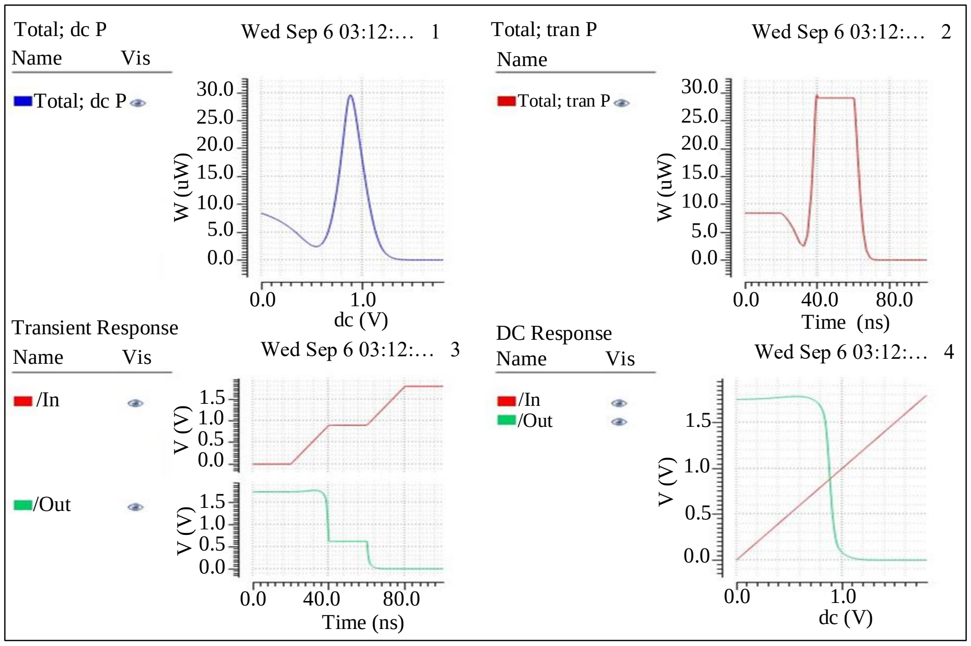


Fig. 8 Results of the proposed low power forced stack ternary inverter

The results of a low-power Ternary inverter using a forced stack approach implemented with a multi-threshold transistor are shown in Figure 8.

Case 3: Ternary Inverter with Resistive Load

A simple ternary inverter with a resistive load is shown in Figure 9. The circuit consists of pMOS, nMOS, resistive load, Vdc=0.9V and 1.8V. When the input voltage is 0V, pMOS is turned ON, and nMOS is turned OFF, pulling the output voltage Vout to Vdc=1.8V. When the input voltage is 0.9V, nMOS turns ON, and the circuit behaves as a voltage divider, yielding output voltage Vout=0.9V. When the input voltage is 1.8V, nMOS turns ON, pulling the output to GND. This circuit uses an additional voltage source 0.9V and a resistor to provide an intermediate logic level. Figure 10 shows the simulation results obtained after implementation. The truth table of the Ternary inverter is satisfied. One drawback of this circuit is a higher leakage power due to a resistive load, as seen in Table 5.

After the implementation of the Ternary inverters with various specifications, it has been observed that the forced stack Ternary inverter with threshold transistors has the lowest leakage power of 582.125fWatt, compared to the Ternary inverter with multi threshold MOSFETs and Ternary inverter with resistive load.

It can be observed that the static leakage power of forced stack Ternary inverters with multi-threshold MOSFETs is 10.87% of Ternary inverters with the highest leakage power dissipation. Hence, this forced stack ternary inverter with multi-threshold transistors, shown in Figure 7, will be used as a Simple Ternary Inverter (STI) in this proposed work's relevant Ternary logic circuit.

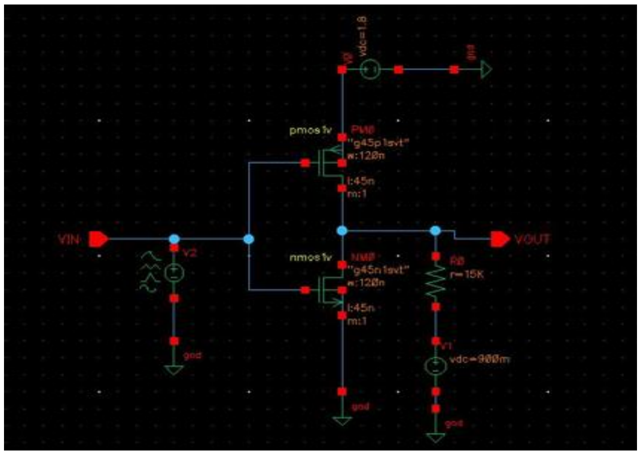


Fig. 9 Ternary inverter with resistive load

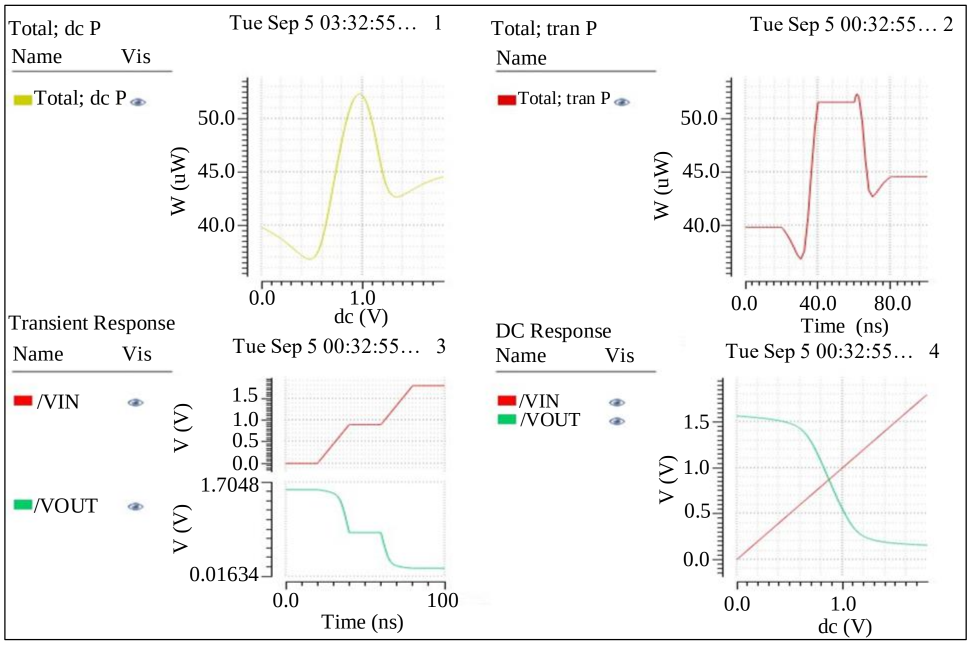


Fig. 10 Results of ternary inverter with resistive load

Table 5. Power analysis of ternary inverters

Type of Inverter	Leakage Power	DC Power	Transient Power
Ternary Inverter with Multi Threshold MOSFETs	975.813 fW	18.1261 µW	19.4573 µW
Forced stack Ternary Inverter with Multi Threshold MOSFETs	582.125 fW	14.27 µW	13.362 µW
Ternary Inverter with Resistive load	5.35313 pW	52.3321 µW	51.5506 µW

3.1.2. Proposed Forced Stack, Multi-Threshold Based Ternary NAND Gate

The Ternary NAND gate implemented with forced stack technique and multi-threshold transistors is shown in Figure 11. The circuit is functionally verified for all the input cases, as described in Table 6.

$$T_{NAND} = \overline{\min\{A, B\}} \tag{4}$$

As per Equation 4, Ternary AND produces minimum among its inputs as the output. Ternary NAND gives the Ternary complimentary of Ternary AND gate. The pMOS network is parallel, and the nMOS network is in a series to obtain the NAND operation. Similarly, the Ternary NOR gate is implemented with a forced stack technique and multi-threshold transistors, as shown in Figure 12. The circuit is functionally verified for all the input cases, as described in Table 6.

$$T_{NOR} = \overline{\max\{A, B\}} \tag{5}$$

As per Equation 5, the Ternary OR gate produces the maximum among its inputs as the output. Ternary NOR gives the Ternary complimentary of Ternary OR gate. The pMOS network is in series, and the nMOS network is in parallel to obtain the NOR operation.

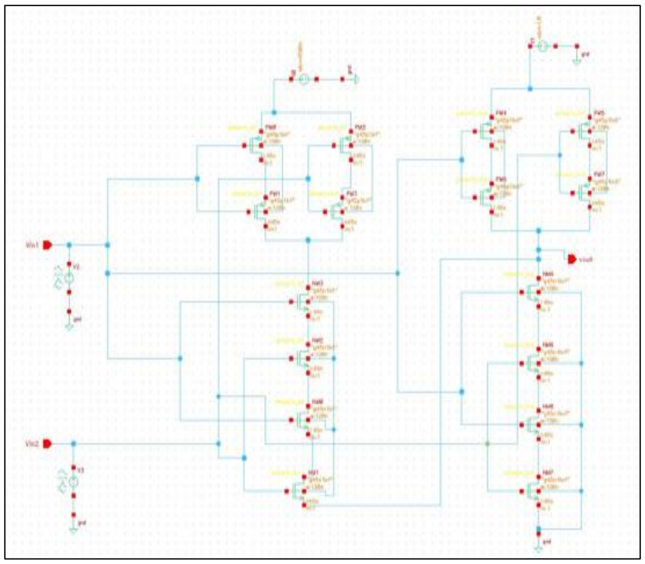


Fig. 11 Proposed low power ternary NAND gate

3.1.3. Proposed Forced Stack, Multi-Threshold Based Ternary NOR Gate

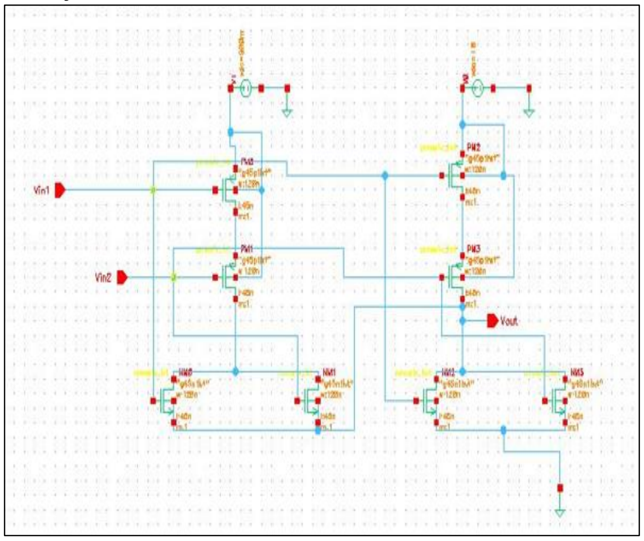


Fig. 12 Proposed low power NOR gate

The truth table for the low-power Ternary NAND and low-power Ternary NOR can be seen in Table 2.

Table 6. Truth table of ternary NAND gate and ternary NOR gate

A	B	T_{NAND}	T_{NOR}
0	0	2	2
0	1	2	1
0	2	2	0
1	0	2	1
1	1	1	1
1	2	1	0
2	0	2	0
2	1	1	0
2	2	0	0

3.1.4. Ternary Decoder

A Ternary decoder uses Positive Ternary Inverter (PTI), Negative Ternary Inverter (NTI) and a Ternary NOR gate as the fundamental building block. Implementing a PTI and NTI with multi-threshold transistors provides the required functionality.

Positive Ternary Inverter (PTI) is shown in Figure 13. It uses a low threshold voltage pMOS (-0.4V) and a high threshold voltage nMOS (0.95V). The truth table of PTI is shown in Table 7. When the input VIN is 0V (Representing Logic 0) or 0.9V (Representing Logic 1), the pMOS is ON, while nMOS is OFF. This pulls the output VOUT to VDC=1.8V, represented as logic 2 in Ternary. When the input VIN is 1.8V, pMOS turns OFF, nMOS turns ON and pulls the output VOUT to 0V. The truth table of PTI, shown in Table 7, is functionally verified.

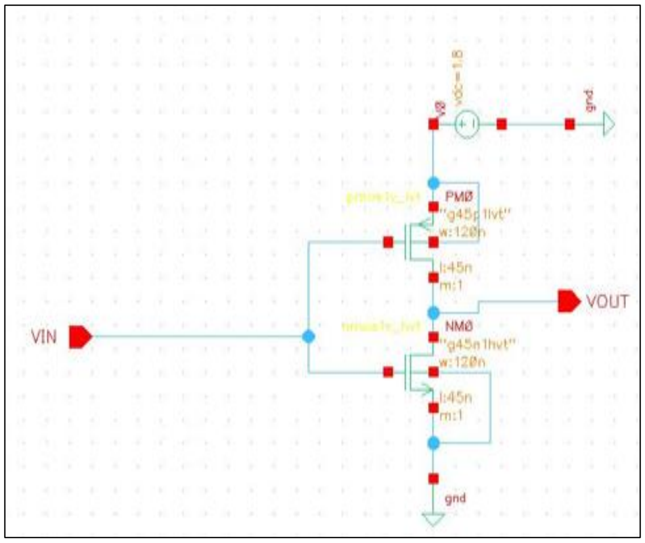


Fig. 13 Positive terminal inverter

Table 7. Truth table of PTI

Input	Output
0	2
1	2
2	0

Negative Terminal Inverter (NTI) uses a high threshold voltage pMOS (Threshold voltage is -0.9V) and a low threshold voltage nMOS (Threshold Voltage is 0.4V) for functioning, as shown in Table 8. The schematic of an NTI is shown in Figure 14.

When the input voltage VIN is 0V, the pMOS turns ON and pulls the output to Vdc=1.8V, representing logic 2. When input voltage VIN is 0.9V or 1.8V, the nMOS turns ON, pulling down the output to Ground, thereby representing logic 0.

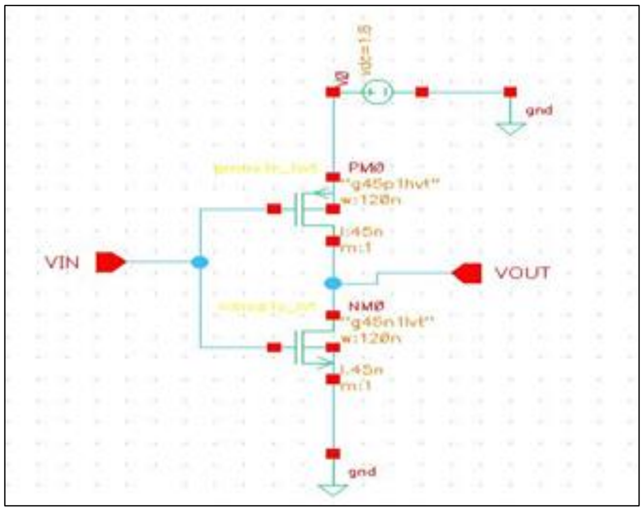


Fig. 14 Negative Threshold Inverter (NTI)

Table 8. Truth table of NTI

Input	Output
0	2
1	0
2	0

The PTI and NTI circuits are symbolically represented and labelled. They are subsequently used in the Ternary decoder shown in Figure 15. The ternary decoder is a combinational circuit with a single input that generates three outputs [12, 26]. It generates unary functions for an input Xk, as shown in Table 9. The response of the ternary decoder to the input X is given by,

$$\mathrm{Xk}=\begin{cases}2, X=k\\0, X\neq k\end{cases} \qquad (6)$$

Where k can take logic values of 0, 1, and 2. The simulation results of the Ternary decoder are shown in Figure 16. Table 9 shows that the decoder has only two logic levels, namely logic 0 and logic 2, corresponding to the highest and lowest logic states, respectively.

Table 9. Truth table of a ternary decoder

Input	V0	V1	V2
0	2	0	0
1	0	2	0
2	0	0	2

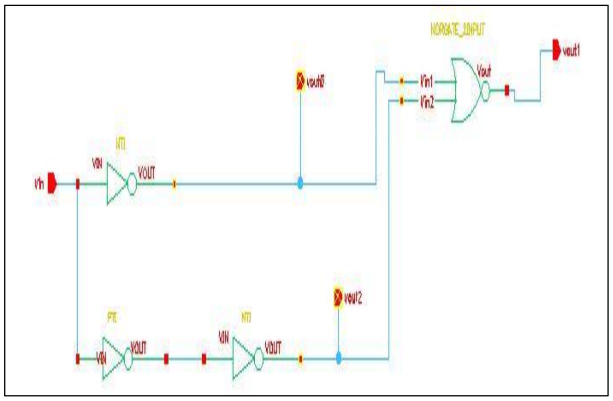


Fig. 15 Ternary decoder

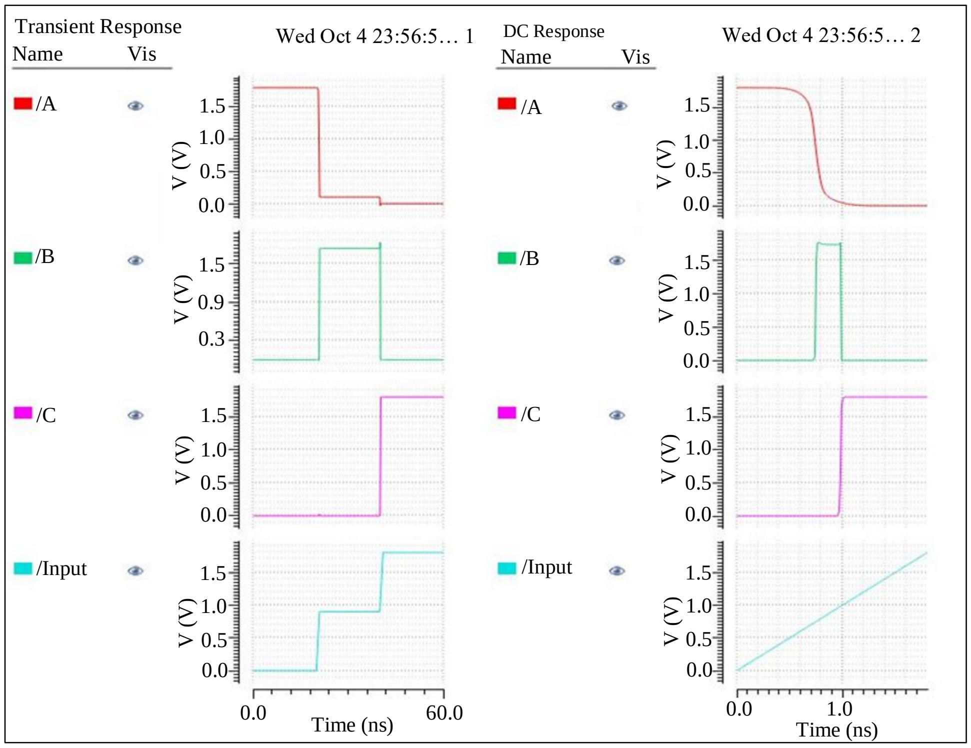


Fig. 16 Results of ternary decoder

3.1.5. Ternary Multiplexer

Ternary decoders are the basic building block of Ternary multiplexers. The 3:1 multiplexer has three inputs: A, B, C, 1 select line S and 1 output Y. The multiplexer equation Y is given by

$$Y = S_0*A+S_1*B+S_2*C \tag{7}$$

S_0, S_1, and S_2 are the select lines corresponding to Ternary values 0, 1, and 2, respectively. These chosen lines are generated from the output of the Ternary decoder. The truth table of a 3:1 multiplexer is shown in Table 10. The gate level circuit of the Ternary multiplexer is shown in Figure 17.

Table 10. Truth table for 3:1 multiplexer

Input	Select Line (S)	Output Y
A	S_0	A
B	S_1	B
C	S_2	C

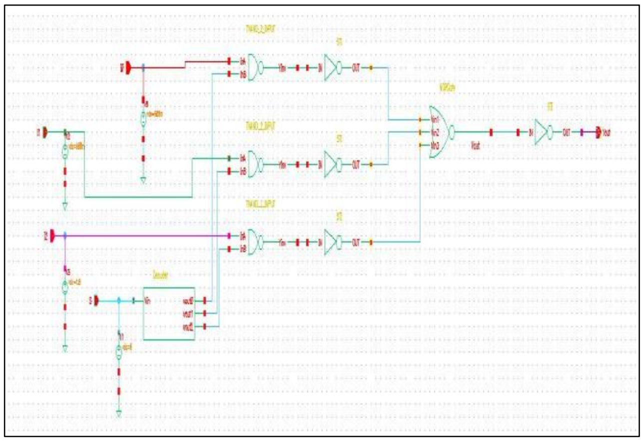


Fig. 17 Ternary multiplexer

The proposed forced stack Ternary gates and digital circuits are used as building blocks for the implementation of Galois field elements and Galois field adder.

4. Methodology

This paper focuses on implementing the Galois field and Galois field modulo adder, as shown in Figure 18. The proposed low-power gates are used for the Galois field implementation and Galois field adder. This work is further extended to provide modular sum in Galois Field (GF) (3^4).

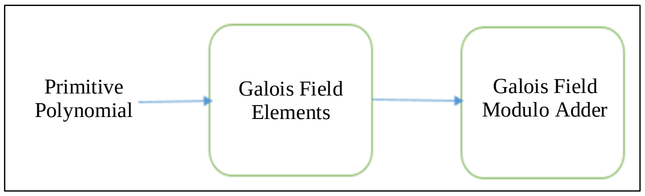


Fig. 18 Block diagram representation of Galois Field modulo adder

4.1. Implementation of Galois Field Elements

This paper focuses on implementing the Galois field using three different designs, which includes:

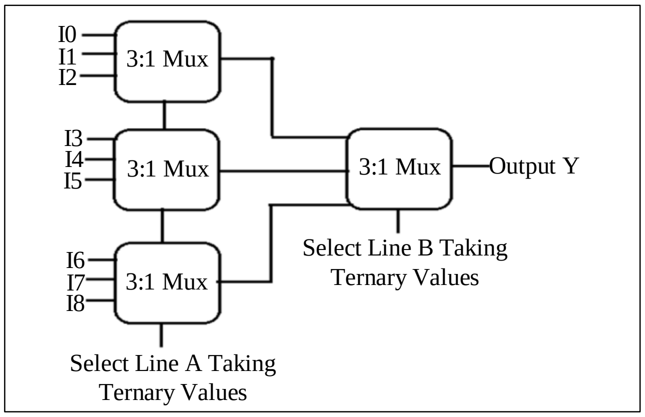
a) Using 3:1 MUX (using Low power gates)
b) Using 9:1 MUX (using low-power gates)
c) Using Canonical Logic expression(using low power gates)

The static power dissipation and the transistor count for the abovementioned cases concerning the conventional gates have been analysed.

4.2. Ternary Galois Field Implementation Using Multiplexer

4.2.1. Case 1

Implementation using a pair of four 3:1 multiplexers for the output Y0 and Y1. The ternary 3:1 multiplexer is shown in Figure 19. Using a 3:1 multiplexer as a fundamental building block, the Galois Field (GF) (3^2) has been implemented. This is pictorially represented in Figure 19. Higher order Galois field can also be implemented using 3:1 multiplexer as a fundamental building block with appropriate select lines. The Ternary Galois Field (GF) (3^2) is implemented using four 3:1 multiplexers, as shown in Figure 19.


I0
I1
I2
3:1 Mux
I3
I4
I5
3:1 Mux
I6
I7
I8
3:1 Mux
3:1 Mux
Output Y
Select Line B Taking Ternary Values
Select Line A Taking Ternary Values
Fig. 19 Construction of 9:1 mux using 3:1 mux

4.2.2. Case 2: Implementation Using Two 9:1 Multiplexers

Each 9:1 multiplexer has nine inputs named I0 to I8, two select lines A and B, taking Ternary input values and an output Y. The logical expression for a 9:1 multiplexer is as shown:

$$Y=A_0*B_0*I_0+A_0*B_1*I_1+A_0*B_2*I_2+A_1*B_0*I_3+A_1*B_1*I_4+A_1*B_2*I_5+A_2*B_0*I_6+\ldots..+A_2*B_1*I_7+A_2*B_2*I_8 \tag{8}$$

The 9:1 multiplexer is implemented using the proposed Ternary AND gate and Ternary OR gate. Ternary decoders are used to generate the select lines of the multiplexer. The working of the same has been verified. A pair of 9:1 multiplexer has been utilised to realise the Galois Field (GF) (3^2). Each element belonging to Galois Field (GF) (3^2) is implemented using its corresponding Ternary value, as shown in Table 11.

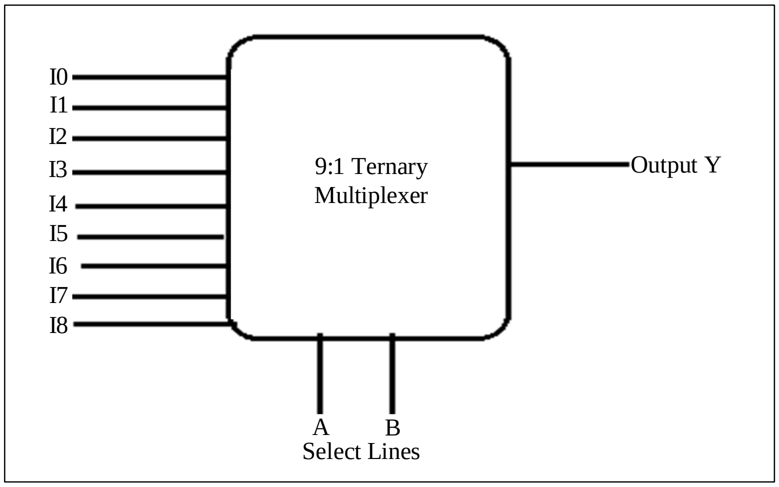


Fig. 20 Symbolic representation of 9:1 multiplexer

Table 11. Ternary representation of GF (3^2)

Galois Field (GF) Elements and its Polynomial Representation	Select Lines	Ternary Representation of GF Elements (Y1)	Ternary Representation of GF Elements (Y0)
0=0	A_0B_0	$I0=(0)_3$	$I0=(0)_3$
1=1	A_0B_1	$I1=(0)_3$	$I1=(1)_3$
α=α	A_0B_2	$I2=(1)_3$	$I2=(0)_3$
$\alpha^2=2\alpha+1$	A_1B_0	$I3=(2)_3$	$I3=(1)_3$
$\alpha_3=2+2\alpha$	A_1B_1	$I4=(2)_3$	$I4=(2)_3$
$\alpha^4=2$	A_1B_2	$I5=(0)_3$	$I5=(2)_3$
$\alpha^5=2\alpha$	A_2B_0	$I6=(2)_3$	$I6=(0)_3$
$\alpha^6=\alpha+2$	A_2B_1	$I7=(1)_3$	$I7=(2)_3$
$\alpha^7=1+\alpha$	A_2B_2	$I8=(1)_3$	$I8=(1)_3$

Case 3: Implementation of Ternary Galois Field (GF) (3^2) Using Canonical Logic Expression

Table 12. Representation of Galois Field (GF) elements

Decimal Number and its Ternary Equivalent	Galois Field (GF) Elements and its Polynomial Representation	Ternary Representation of GF Elements (Y1)	Ternary Representation of GF Elements(Y0)
0=(00)	0=0	$(0)_3$	$(0)_3$
1=(01)	1=1	$(0)_3$	$(1)_3$
2=(02)	α=α	$(1)_3$	$(0)_3$
3=(10)	$\alpha^2=2\alpha+1$	$(2)_3$	$(1)_3$
4=(11)	$\alpha_3=2+2\alpha$	$(2)_3$	$(2)_3$
5=(12)	$\alpha^4=2$	$(0)_3$	$(2)_3$
6=(20)	$\alpha^5=2\alpha$	$(2)_3$	$(0)_3$
7=(21)	$\alpha^6=\alpha+2$	$(1)_3$	$(2)_3$
8=(22)	$\alpha^7=1+\alpha$	$(1)_3$	$(1)_3$

Let (x, y) represent the Ternary equivalent of a decimal number. As an illustration, decimal 0 is represented as X_0Y_0. Similarly, decimal 1 is represented as X_0Y_1.

Hence, the logical expression for $Y1=\sum m(3,4,6)_{logic2} + \sum m\,(2,7,8)_{logic1}$:

$$Y1 = (X_1Y_0+X_1Y_1+X_2Y_0)_{(LOGIC2)} + (X_0Y_2+X_2Y_1+X_2Y_2)_{(LOGIC1)} \quad (9)$$

The expression Y1 is implemented using low power NAND gates, as shown in Figure 21.

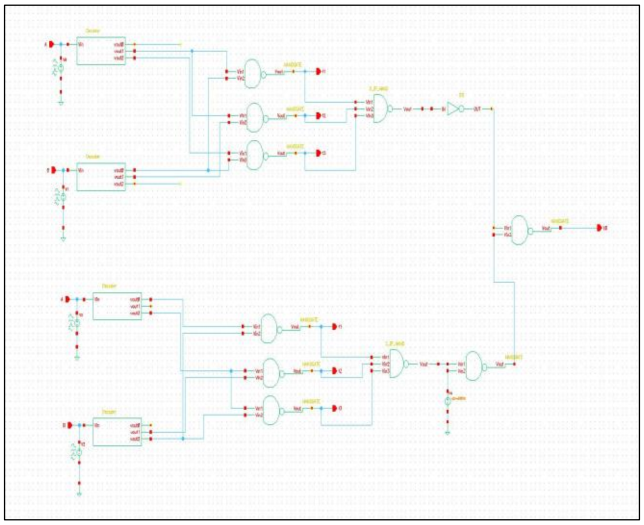


Fig. 21 Implementation of expression Y1 using low power NAND gates

The logical expression for $Y0=\sum m(4,5,7)_{logic2} + \sum m\,(1,3,8)_{logic1}$:

$$Y0= (X_1Y_1+X_1Y_2+X_2Y_1)_{(LOGIC2)} + (X_0Y_1+X_1Y_0+X_2Y_2)_{(LOGIC1)} \quad (10)$$

This expression is realised using the proposed Ternary NAND gates similar to the implementation shown in Figure 21. The implementation of the Galois field using multiplexers and canonical logic expression is functionally verified. These two different Ternary Galois field implementation methods are compared concerning the number of transistors used and the leakage power. The comparison results are tabulated in Table 13.

It has been observed that implementing the Galois field using the canonical logic expression provides a low leakage power dissipation. It can be observed that the Galois field implementation using the Canonical Logic expression has a static power dissipation of 52% compared to the highest leakage power dissipation, as shown in Table 13. These generated field elements are fed to the proposed Galois field adder, elaborated in the succeeding section.

Table 13. Comparison of various implementations of the Galois Field

Galois Field Implementation Using	Transistor Count	Leakage Power
3:1 MUX (Using Conventional Gates)	396	197.1414 μW
3:1 MUX (Using the Proposed Forced Stack Technique)	792	65.7138 μW
9:1 MUX	416	212.2516 μW
Canonical Logic Expression (Using Conventional Gates)	256	110.376 μW
Canonical Logic Expression (Using Low Low-Power Gates)	512	36.792 μW

4.3. Galois Field Adder

Modular arithmetic is popularly used in most cryptographic and error-control coding applications. This research work focuses on modular addition operation over the Ternary Galois field. Table 14 illustrates the addition operation over two Ternary values.

Table 14. Modular addition

A	B	Modular Sum
0	0	0
0	1	1
0	2	2
1	0	1
1	1	2
1	2	3 mod 3=0
2	0	2
2	1	3 mod 3=0
2	2	4 mod 3=1

In this research work, the Galois field adder is implemented using two different design methods, such as:

1. Modulo adder using canonical logic expression.
2. Modulo adder uses a novel approach.

These implementation methods concerning the gate count and the number of transistors with the conventional Ternary Half adder have been analysed.

4.3.1. Case 1: Implementation of Modular Sum Using Canonical Logic Expression

The canonical logic expression for the modular sum is,

$$(A_0B_1+A_1B_0+A_2B_2)_{(LOGIC1)}+(A_0B_2+A_1B_1+A_2B_0)_{(LOGIC2)} \quad (11)$$

The logic expression for obtaining modular sum is implemented, and the truth table is verified.

4.3.2. Case 2: A Novel Approach for the Implementation of Modular Sum

The novel approach for implementing the Ternary modular sum is shown in Table 15. As an illustration, let us consider the inputs to be added as A= $(1)_3$ and B= $(1)_3$. The modular sum is 2, as shown in Table 15. The Ternary XOR gate gives an output of 1 for the inputs A= $(1)_3$ and B= $(1)_3$. The Ternary XOR gate output of $(1)_3$ is fed to a Positive Threshold Inverter (PTI) (Table 5). The output obtained is $(2)_3$. Thus, the modular sum is obtained by optimising the circuit with a single Ternary XOR gate, two PTI, Two NTI and one Ternary AND gate. Low-power Ternary XOR gates are used to implement modular addition. These Ternary XOR gates are built using the proposed low power NAND gates, as seen in Figure 11, with a forced stack approach. The symbolic representation of the proposed Low-power NAND gate is used to construct the Ternary XOR gate.

Table 15. Simplifications for obtaining modular sum

A	B	Results of a Ternary XOR Gate	Modular Sum	Optimised Expression for Modular Sum
0	0	0	0	A⊕B
0	1	1	1	A⊕B
0	2	2	2	A⊕B
1	0	1	1	A⊕B
1	1	1	2	$\overline{A\oplus B}$; Inverter used is Positive Threshold Inverter (PTI)
1	2	1	3 mod 3=0	$\overline{A\oplus B}$;; Inverter used is a Negative Threshold Inverter (NTI)
2	0	2	2	A⊕B
2	1	1	3 mod 3=0	$\overline{A\oplus B}$;; Inverter used is a Negative Threshold Inverter (NTI)
2	2	0	4 mod 3=1	(($\overline{A\oplus B}$)●log ic1); Inverter used is a Positive Threshold Inverter (PTI).

Ternary XOR Gate

The 2-input Ternary XOR gate is implemented using the expression Y=$\bar{A}B+A\bar{B}$ with the proposed power NAND Gates, as shown in Figure 22. The functional truth table of the Ternary XOR gate is shown in Table 16. The truth table of the Ternary XOR gate is functionally verified for all the possible inputs.

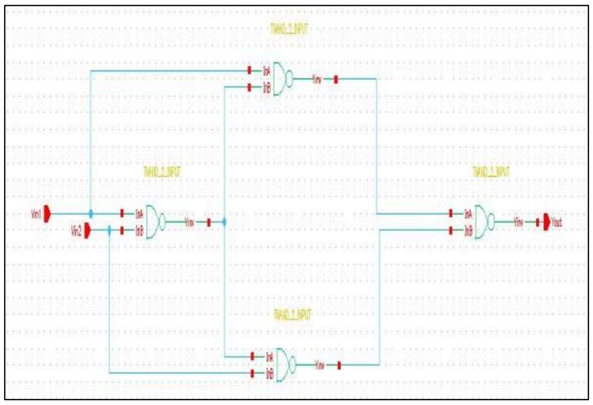


Fig. 22 Low-power ternary XOR gate

Table 16. Functional truth table of ternary XOR gate

A	B	Y
0	0	0
0	1	1
0	2	2
1	0	1
1	1	1
1	2	1
2	0	2
2	1	1
2	2	0

Figure 23 shows the results of the Ternary XOR gate for a few cases of input. Considering the truth table of the Galois field modulo adder and the Ternary XOR gate in Table 14 and Table 16, respectively, specific simplification techniques are adopted to obtain the modular sum that results in an optimised expression, as seen in Table 15.

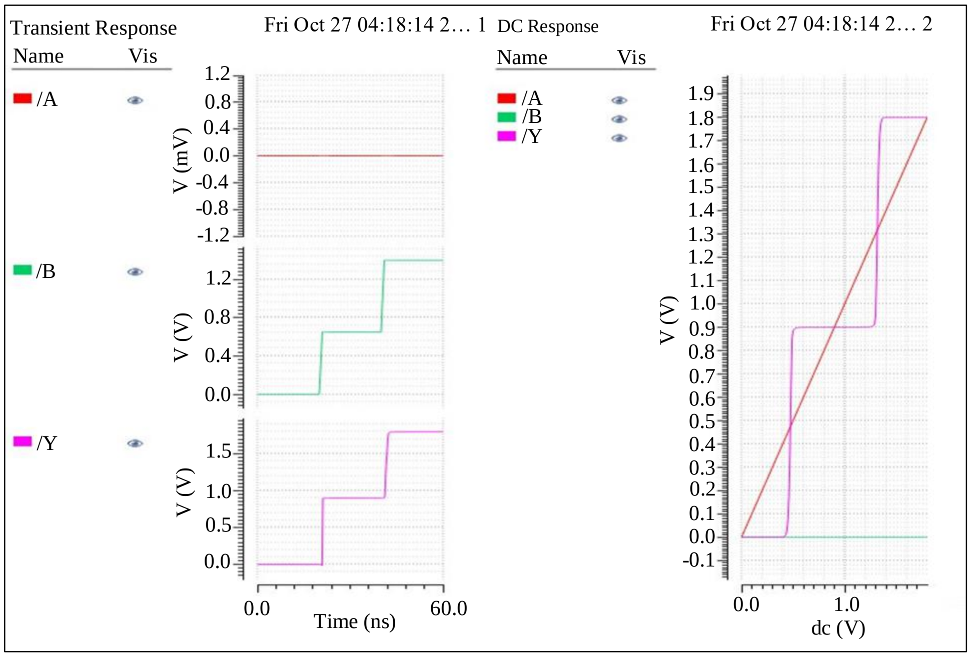


Fig. 23 Results of ternary XOR gate

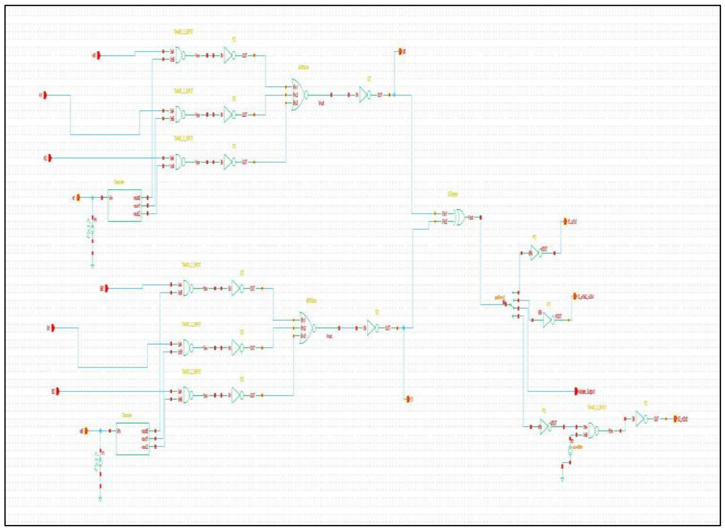


Fig. 24 Modular adder

The Results of the Ternary XOR gate and the modular sum are compared to obtain the optimised expression. The logical expression is implemented using Cadence Virtuoso using GPDK045. The logical diagram can be seen in Figure 24. The logic diagram for a modular adder can be seen in Figure 24. The logical diagram consists of a Ternary decoder, Ternary XOR gate, Positive Ternary Inverter (PTI), Negative Ternary Inverter (NTI), Ternary AND gate and switch. The decoder is used to generate the various Ternary inputs. The Ternary XOR gate is the primary logic gate to obtain the modulo sum output. The PTI and NTI are used to create the modulo sum for specific input cases, as illustrated in Table 13, which are switch-controlled. The modulo adder is functionally verified for all the test cases. Figure 25 shows the results obtained from all the various combinations of input.

As an Illustration of Galois field, adder:
Let us consider addition of $A=\alpha^3$ and $B=\alpha^4$, over GF (3^2)
Ternary Representation of A= $(22)_3$ and B= $(02)_3$
Their Sum= $(22)_3+(02)_3=(21)_3=\alpha^2$

The proposed low-power Galois field adder using the canonical expression, the Ternary half adder circuit and the proposed novel method of implementing the modulo adder are compared with the number of gates and transistors used, as shown in Table 17.

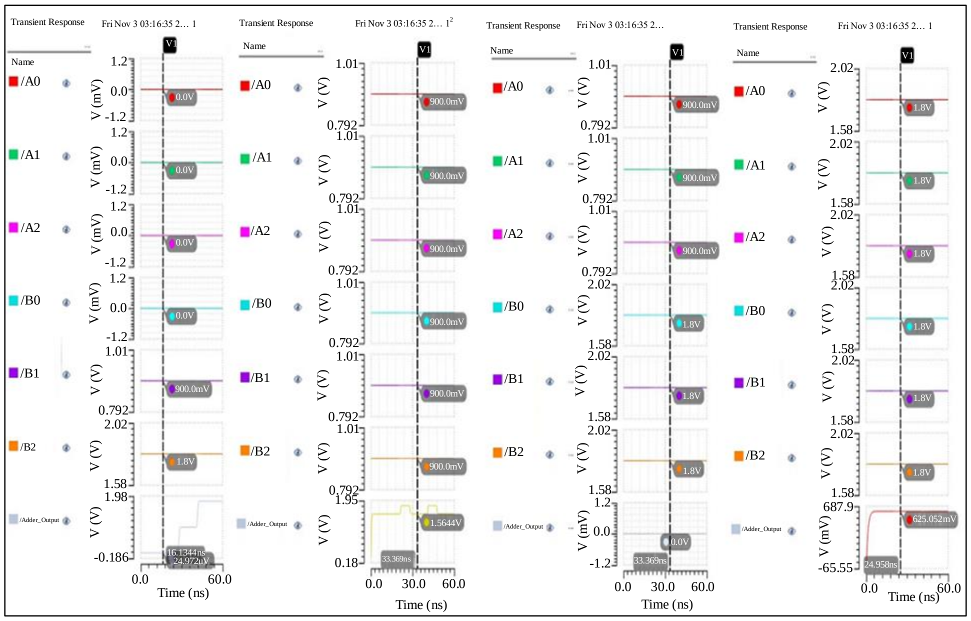


Fig. 25 Results of modular adder for all the test cases

Table 17. Comparison of conventional adder and modulo adder

	Implementation of Power Galois Field Adder Using the Canonical Logic Expression (Circuit 1)	Low Power Half Adder Circuit [27, 28] (Circuit 2)	Novel Method of Implementing Low Power Galois Field Adder
No. of Gates	2 Decoders, 10 NAND Gates and 2 Inverters	2 Decoders, 14 NAND Gates, Two Inverters	2 Decoders, 1 XOR Gate(4 NAND Gates), 2 PTI and 2 NTI, 1 AND Gate
No of Transistors	204 Transistors	268 Transistors	124 Transistors

The comparison results tabulated in Table 17 show that the proposed method of implementing the Galois field adder is optimised concerning the number of transistors used and has less leakage power. Further, it has been analysed that the proposed novel method of implementing the Galois field adder consumes only 46% of the transistor count compared to the conventional Ternary half adder and 60.78% of the transistor count compared to the implementation using Canonical logic expression.

Since all the logic circuits are implemented using the forced stack technique with multi-threshold transistors, there is a low static power dissipation. Hence, the proposed novel method of implementing the Galois field adder is considered and extended to add the higher-order Ternary Galois field data.

5. Results and Discussion

The proposed novel implementation of modulo adder is used as a component and instantiated to perform addition on higher order Ternary Galois Field. This research is extended for the implementation of,

1. GF (3^4), using the canonical logic expression.
2. GF (3^4) modulo adder.

5.1. Case 1: Implementation of GF (3^4) Using Canonical Logic Expression

In this research work, modulo adder for GF (3^4) is implemented. There are a total of 81 elements in GF (3^4). They are $0,1,\alpha,\alpha^2,\alpha^3,\ldots\ldots\ \alpha^{79}$.

To construct the elements of the Galois Field (GF) (3^4), the primitive polynomial $p(x)=2+x+x^4$ is used.

Let α be a root of the polynomial p(x).

$P(x==\alpha)=2+\alpha+\alpha^4=0$

$\alpha^4=-2-\alpha$

$$\alpha^4=1+2\alpha \tag{12}$$

The elements of GF (3^4), its polynomial representation, the Ternary representation and the decimal representation are shown in Table 18 (Appendix).

The canonical logic expression is derived using the method discussed in Table 12 and the Equation 9. According to the same method, the obtained canonical logic expression for Y0 is shown in Equation 12. The expression for Y0 using the minterms is,

$$Y0= \sum^{m(1,5,9,11,12,13,20,....)logic_{\_1}}_{\sum m(8,14,17,18,22,23.....)logic\_2} + \tag{13}$$

Let us assume (w, x, y, z) represent the Ternary input, with ‘w’ carrying the highest Ternary weightage and ‘z’ having the least Ternary weightage. The canonical logic expression for Y0 is,

$$Y0=(W0X0Y0Z1+W0X0Y1Z2+W0X1Y0Z0+W0X1Y0Z2+W0X1Y1Z0+ W0X1Y1Z1+W0X2Y0Z2+.................)\log ic\_1+(W0X0Y2Z2+W0X1Y1Z2+W0X1Y2Z2 +W0X2Y0Z0+.................)\log ic\_2. \tag{14}$$

Similarly, the canonical expression for Y1, Y2, and Y3 is expressed using minterms and the input variables w, x, y, and z. These canonical logic expressions are implemented using the proposed low power Ternary gates. These Galois field elements are fed as inputs to the modulo adder shown in Figure 25.

5.2. Case 2: Implementation of GF (3^4) Modulo Adder

Galois Field (GF) (3^4) is implemented using the canonical Logic expression, and the output is obtained with fewer gates and transistors. It also results in low leakage power dissipation, as the proposed Ternary gates use multi-threshold transistors. These values are deployed for implementing modulo adder, as shown in Figure 26.

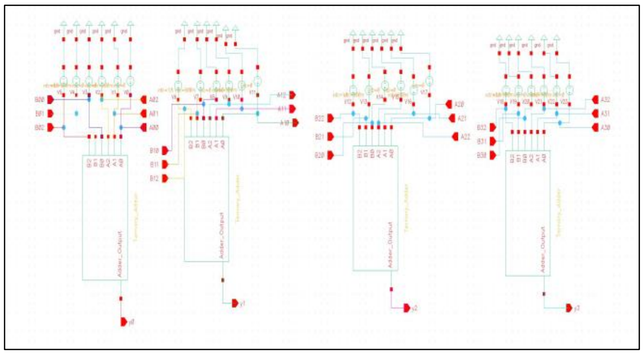


Fig. 26 Galois Field (GF) (3^4) adder

The symbolic representation of the proposed modulo adder discussed in the previous section is used as a component. It is instantiated for constructing the Galois Field (GF) (3^4) in the structural modelling.

As an illustration, let us consider the addition of two elements over GF (3^4):

$$A=\alpha^5 \ \& \ B=\alpha^6$$

The Ternary representation of A= $(0120)_3$ and B= $(0012)_3$ as per Table 17. The modular sum of A and B over GF (3^4) is $\frac{\overset{0120}{\underline{0012}}}{0102}$ that is (0102) $=\alpha^{58.}$

These results are verified using the modular adder shown in Figure 25. Several input vectors are fed to the circuit for testing, and the modulo sum is successfully verified. Higher-order Galois field adders can be implemented and verified using the component instantiation approach in structural modelling.

6. Conclusion

After implementing the Ternary logic gates with different techniques (Table 5), the forced stack technique using multi-threshold transistors has been shown to have the lowest leakage power.

Hence, Ternary decoders and multiplexers are implemented using low power Ternary logic gates. Four different types of Ternary Galois field implementation have been successfully carried out. Table 10 shows that the realisation using logic expression offers a 56% reduction in the leakage power.

This Ternary Galois field implementation is further deployed to build a modular adder circuit. Implementing a novel modulo adder using an XOR gate reduces the transistor count. This technique proposed consumes 60% and 46% of the chip area compared to circuit 1 and circuit 2, respectively, as shown in Table 17. The abovementioned implementation is extended for Galois Field (GF) (3^4). The same can be further extended to design and realise a modular multiplier for cryptographic processors.

References

[1] Subrata Das, Partha Sarathi Dasgupta, and Samar Sensarma, “Arithmetic Algorithms for Ternary Number System,” *Progress in VLSI Design and Test, 16th International Symposium on VSLI Design and Test*, pp. 111-120, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[2] A.P. Dhande, V.T. Ingole, and V.R. Ghiye, *Ternary Digital System: Concepts and Applications*, SM Online Publishers LLC, pp. 1-131, 2014. [Google Scholar] [Publisher Link]

[3] Malachy Eaton, “Design and Construction of a Balanced Ternary ALU with Potential Future Cybernetic Intelligent Systems Applications,” *2012 IEEE 11th International Conference on Cybernetics, Intelligent Systems (CIS)*, Limerick, Ireland, pp. 30-35, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[4] Xiao-Yuan Wang et al., "A Review on the Design of Ternary Logic Circuits," *Chinese Physics B*, vol. 30, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] B. Srinivasu, and K. Sridharan, "A Synthesis Methodology for Ternary Logic Circuits in Emerging Device Technologies," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 8, pp. 2146-2159, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[6] A.P. Dhande, Satish S. Narkhede, and Shridhar S. Dudam, "VLSI Implementation of Ternary Gates Using Tanner Tool," *2014 2nd International Conference on Devices, Circuits and Systems (ICDCS)*, Coimbatore, India, pp. 1-5, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[7] Sneh Lata Murotiya, Anu Gupta, and Ayan Pandit, "CNTFET-Based Low Power Design of 4-Input Ternary XOR Function," *2014 International Conference on Computer and Communication Technology (ICCCT)*, Allahabad, India, pp. 347-350, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[8] V.T. Gaikwad, and P.R. Deshmukh, "Design of CMOS Ternary Logic Family Based on Single Supply Voltage," *2015 International Conference on Pervasive Computing (ICPC)*, Pune, India, pp. 1-6, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[9] Ahmet Unutulmaz, and Cem Ünsalan, "Implementation and Applications of a Ternary Threshold Logic Gate," *Circuits, Systems, and Signal Processing*, vol. 43, pp. 1192-1207, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Y. Sujatha, and K.N.V.S. Vijaya Lakshmi, "Applications of XOR Gate Using Ternary Logic," *International Journal of Creative Research Thoughts*, vol. 5, no. 4, pp. 2450-2454, 2017. [Publisher Link]

[11] Mingqiang Huang et al., "Design and Implementation of Ternary Logic Integrated Circuits by Using Novel Two-Dimensional Materials," *Applied Sciences*, vol. 9, no. 20, pp. 1-13, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] Vallabhuni Vijay et al., "Design of Unbalanced Ternary Logic Gates and Arithmetic Circuits," *Journal of VLSI Circuits and Systems*, vol. 4, no. 1, pp. 20-26, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Makani Nailesh Kishor, and Satish S. Narkhede, "A Novel FinFET-Based Approach for the Realisation of Ternary Gates," *ICTACT Journal on Microelectronics*, vol. 2, no. 2, pp. 254-260, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[14] Tabassum Khurshid, and Vikram Singh, "Energy Efficient Design of Unbalanced Ternary Logic Gates and Arithmetic Circuits Using CNTFET," *AEU - International Journal of Electronics and Communications*, vol. 163, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] P.A. Gowrisankar, "Design of Multi-Valued Ternary Logic Gates Based on Emerging Sub-32nm Technology," *2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, Chennai, India, pp. 1023-1031, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[16] Badugu Divya Madhuri, and Subramani Sunithamani, "Design of Ternary Logic Gates and Circuits Using GNRFETs," *IET Circuits, Devices & Systems*, vol. 14, no. 7, pp. 972-979, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Zarin Tasnim Sandhie, Farid Uddin Ahmed, and Masud H. Chowdhury, "Design of Ternary Logic and Arithmetic Circuits Using GNRFET," *IEEE Open Journal of Nanotechnology*, vol. 1, pp. 77-87, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Kushawaha Jyoti, and Satish Narkhede, "An Approach to Ternary Logic Gates Using FinFET," *AICTC' 16 Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, pp. 1-6, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[19] Sunmean Kim, Taeho Lim, and Seokhyeong Kang, "An Optimal Gate Design for the Synthesis of Ternary Logic Circuits," *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jeju, Korea (South), pp. 476-481, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[20] A. Steegen et al., "65nm CMOS Technology for Low Power Applications," *IEEE International Electron Devices Meeting, 2005. IEDM Technical Digest*, Washington, USA, pp. 64-67, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[21] K. Koh et al., "Highly Manufacturable 100nm 6T Low Power SRAM with Single Poly-Si Gate Technology," *2003 International Symposium on VLSI Technology, Systems and Applications*, Hsinchu, Taiwan, pp. 64-67, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[22] S. Zhao et al., "Transistor Optimisation for Leakage Power Management in a 65nm CMOS Technology for Wireless and Mobile Applications," *2004 Symposium on VLSI Technology*, Honolulu, USA, pp. 14-15, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[23] C.H. Kim et al., "PVT-Aware Leakage Reduction for On-Die Caches with Improved Read Stability," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 170-178, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[24] K. Flautner et al., "Drowsy Caches: Simple Techniques for Reducing Leakage Power," *Proceedings 29th Annual International Symposium on Computer Architecture*, Anchorage, USA, pp. 148-157, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[25] Aloke Saha, Narendra Deo Singh, and Dipankar Pal, "Efficient Ternary Comparator on CMOS Technology," *Microelectronics Journal*, vol. 109, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[26] Ramzi A. Jaber et al., "CNFET-Based Designs of Ternary Half-Adder Using a Novel 'Decoder-less' Ternary Multiplexer Based on Unary Operators," *Microelectronics Journal*, vol. 96, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[27] Rituraj Yadav, Ashish Sura, and Sunita Dahiya, "Half Adder Design Using Various Technologies and Comparison of Various Parameter Performance," *International Journal of Engineering Applied Sciences and Technology*, vol. 6, no. 2, pp. 95-100, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[28] Jihad Mohamed Aljaam, Ramzi A. Jaber, and Somaya Ali Al-Maadeed, "Novel Ternary Adder and Multiplier Designs Without Using Decoders or Encoders," *IEEE Access*, vol. 9, pp. 56726-56735, 2021. [CrossRef] [Google Scholar] [Publisher Link]

Appendix

Table 18. Galois Field (GF) (3^4) representation

Elements	Polynomial Representation	Vector Form	Decimal Representation
		Ternary Representation = $\alpha^0\alpha^1\alpha^2\alpha^3$	**Weightage=$3^0 3^1 3^2 3^3$**
0	0	$(0000)_3$	0
1	1	$(1000)_3$	1
α	α	$(0100)_3$	3
α^2	α^2	$(0010)_3$	9
α^3	α^3	$(0001)_3$	27
α^4	$1+2\alpha$	$(1200)_3$	7
α^5	$\alpha+2\alpha^2$	$(0120)_3$	19
α^6	$\alpha+2\alpha^3$	$(0012)_3$	63
α^7	$2+\alpha+\alpha^3$	(2102)3	59
α^8	$1+\alpha+\alpha^2$	(1110)3	13
α^9	$\alpha+\alpha^2+\alpha^3$	(0111)3	39
α^{10}	$1+2\alpha+\alpha^2+\alpha^3$	(1211)3	43
α^{11}	$1+2\alpha^2+\alpha^3$	(1021)3	46
α^{12}	$1+2\alpha^3$	(1002)3	55
α^{13}	$2+2\alpha$	(2200)3	8
α^{14}	$2\alpha+2\alpha^2$	(0220)3	24
α^{15}	$2\alpha^2+2\alpha^3$	(0022)3	72
α^{16}	$2+\alpha+2\alpha^3$	(2102)3	59
α^{17}	$2+\alpha^2$	(2010)3	11
α^{18}	$2\alpha+\alpha^3$	(0201)3	33
α^{19}	$1+2\alpha+2\alpha^2$	(1220)3	25
α^{20}	$\alpha+2\alpha^2+2\alpha^3$	(0122)3	66
α^{21}	$2+\alpha+\alpha^2+2\alpha^3$	(2112)3	68
α^{22}	$2+\alpha^2+\alpha^3$	(2011)3	38
α^{23}	$1+\alpha+\alpha^3$	(1101)3	31
α^{24}	$1+\alpha^2$	(1010)3	10
α^{25}	$\alpha+\alpha^3$	(0101)3	30

α^{26}	$1+2\alpha+\alpha^2$	(1210)3	16
α^{27}	$\alpha +2\alpha^2+\alpha^3$	(0121)3	48
α^{28}	$1+2\alpha+\alpha^2+2\alpha^3$	(1212)3	43
α^{29}	$2+2\alpha+2\alpha^2+\alpha^3$	(2221)3	53
α^{30}	$1+\alpha+2\alpha^2+\alpha^3$	(1121)3	49
α^{31}	$2+2\alpha+\alpha^2+2\alpha^3$	(2212)3	71
α^{32}	$2+2\alpha^2+\alpha^3$	(2021)3	47
α^{33}	$1+\alpha+2\alpha^3$	(1102)3	58
α^{34}	$2+2\alpha+\alpha^2$	(2210)3	17
α^{35}	$2\alpha+2\alpha^2+\alpha^3$	(0221)3	51
α^{36}	$1+2\alpha+2\alpha^2+2\alpha^3$	(1222)3	79
α^{37}	$2+2\alpha+2\alpha^2+2\alpha^3$	(2222)3	80
α^{38}	$2+2\alpha^2+2\alpha^3$	(2022)3	74
α^{39}	$2+2\alpha^3$	(2002)3	56
α^{40}	2	(2000)3	2
α^{41}	2α	(0200)3	6
α^{42}	$2\alpha^2$	(0020)3	18
α^{43}	$2\alpha^3$	(0002)3	54
α^{44}	$2+\alpha$	(2100)3	5
α^{45}	$2\alpha+\alpha^2$	(0210)3	15
α^{46}	$2\alpha^2+\alpha^3$	(0021)3	45
α^{47}	$1+2\alpha+2\alpha^3$	(1202)3	61
α^{48}	$2+2\alpha+2\alpha^2$	(2220)3	26
α^{49}	$2\alpha+2\alpha^2+2\alpha^3$	(0222)3	78
α^{50}	$2+\alpha+2\alpha^2+2\alpha^3$	(2122)3	77
α^{51}	$2+\alpha^2+2\alpha^3$	(2012)3	65
α^{52}	$2+\alpha^3$	(2001)3	29
α^{53}	$1+\alpha$	(1100)3	4
α^{54}	$\alpha+\alpha^2$	(0110)3	12
α^{55}	$\alpha^2+\alpha^3$	(0011)3	36
α^{56}	$1+2\alpha+\alpha^3$	(1201)3	34
α^{57}	$1+2\alpha^2$	(1200)3	7
α^{58}	$\alpha+2\alpha^3$	(0102)3	57
α^{59}	$2+\alpha+\alpha^2$	(2110)3	14
α^{60}	$2\alpha+\alpha^2+\alpha^3$	(0211)3	42

α^{61}	$1+2\alpha+2\alpha^2+\alpha^3$	(1221)3	52
α^{62}	$1+2\alpha^2+2\alpha^3$	(1022)3	73
α^{63}	$2+2\alpha+2\alpha^3$	(2202)3	62
α^{64}	$2+2\alpha^2$	(2020)3	20
α^{65}	$2\alpha+2\alpha^3$	(0202)3	60
α^{66}	$2+\alpha+2\alpha^2$	(2120)3	23
α^{67}	$2\alpha+\alpha^2+2\alpha^3$	(0212)3	69
α^{68}	$2+\alpha+2\alpha^2+2\alpha^3$	(2122)3	77
α^{69}	$1+\alpha+\alpha^2+2\alpha^3$	(1112)3	67
α^{70}	$2+2\alpha+\alpha^2+\alpha^3$	(2211)3	44
α^{71}	$1+\alpha+2\alpha^2+\alpha^3$	(1121)3	49
α^{72}	$1+\alpha^2+2\alpha^3$	(1012)3	64
α^{73}	$2+2\alpha+\alpha^3$	(2201)3	35
α^{74}	$1+\alpha+2\alpha^2$	(1120)3	22
α^{75}	$\alpha+\alpha^2+2\alpha^3$	(0112)3	66
α^{76}	$2+\alpha+\alpha^2+\alpha^3$	(2111)3	41
α^{77}	$1+\alpha+\alpha^2+\alpha^3$	(1111)3	40
α^{78}	$1+\alpha^2+\alpha^3$	(1011)3	37
α^{79}	$1+\alpha^3$	(1001)3	28
α^{80}	1		1