Design of a Power-Efficient Carbon Nanotube Field Effect Transistor (CNFET)-Based Ternary Logic Processor

THESIS

Submitted in partial fulfillment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Sharvani Gadgil

ID No. 2018PHXF0434H

Under the Supervision of

Dr. Chetan Kumar V

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

2023

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the thesis titled Design of a Power-Efficient Carbon Nanotube Field Effect Transistor (CNFET)-based Ternary Logic Processor, submitted by Sharvani Gadgil ID No 2018PHXF0434H for award of PhD of the Institute embodies original work done by her under my supervision.

[illegible]

Signature of the Supervisor

Name: Dr. Chetan Kumar V

Designation: Assistant Professor

Date: 27.12.2023

DECLARATION

I, Sharvani Gadgil, declare that this written submission represents my ideas in my own words, and where others' ideas or words have been included; I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated, or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action as per the rules and regulations of the Institute.

Sharvani Gadgil

Acknowledgments

The culmination of this thesis is attributed to a diverse group of people who have guided and supported me throughout and together with me have persisted on a challenging journey. At the onset, I would like to sincerely thank my supervisor Dr. Chetan Kumar V for being a true source of inspiration and a guiding light for this research. I would also like to thank my DAC members Dr. Syed Ershad Ahmed and Dr. Soumya J for their time and constructive feedback that helped me at every important milestone.

I express my deep sense of gratitude to BITS-Pilani, Hyderabad campus for the state of the art infrastructure which made sure the research gets all the necessary support required. In the same vein, I would like to thank the Department of EEE for their participation as a cohesive unit at various junctures. I am thankful to Goli Naga Sandesh for the help I received in the technicalities of my work.

Last but not the least, I cannot appreciate enough my daughter Sharanya who despite being a toddler understood that I am occupied with something important and supported me at the same level as my husband Rohit and my two mothers (mother-Anjali and mother-in-law-Neeta). Finally, I dedicate this work to my father Dr. S. B. Gadgil who has inspired me everyday and has been providing me the moral support throughout my PhD journey.

Hyderabad Sharvani Gadgil

Abstract

The interest in Multi-Valued Logic (MVL) circuits has grown over the years in view of the benefits they provide in terms of reducing the complexity of interconnects and increasing information density. Ternary logic is a subset of MVL that uses three logic levels for computation. Implementation of Ternary logic circuits necessitates transistors with different threshold voltages. This is acheived in the traditional Metal-Oxide-Semiconductor (MOS) transistors using body biasing making the design of ternary logic circuits using MOS transistors more complex. Carbon-Nanotube-Field-Effect-Transistor (CNFET) is a good alternative for ternary logic implementation where, transistors with different threshold voltages can be obtained by varying the physical dimensions of the Carbon-Nano-Tube (CNT), which acts as a conduction channel in the CNFETs.

The existing work on CNFET-based ternary logic circuit implementation focuses only on individual blocks like adders, multipliers, ALUs, memory etc but not the entire system. Hence, the primary goal of this thesis is to design a CNFET-based Ternary Logic Processor (TLP). The processor implementation requires the design of different ternary combinational and sequential logic circuits as well as ternary memory.

In this work, firstly , we present a new CNFET-based ternary Arithmetic-Logic-Unit (ALU) architecture. The main components of the ternary ALU are the function select block that the selects the required function the ALU should perform , the transmission gate block that connects the inputs of the ALU to its functional blocks, and the functional modules. The functional modules, which implement the arithmetic and logic functions, like the ternary adder-subtractor, multiplier and the comparator of the ternary ALU are designed using 2 : 1 multiplexers to achieve low power consumption.

The second contribution of this thesis, is novel design methodologies for implementing CNFET-based ternary sequential logic circuits, which form an integral part of a processor. In this work, different designs for D-flipflop are presented. The first flipflop design is implemented using the ternary buffer and Standard Ternary Inverter (STI). The second design is a multiplexer-based design implemented using successor-predecessor circuits. The third and fourth D-flipflop designs are hybrid designs which are a combination of buffer-based and successor-predecessor-based designs. These flipflops are further used to implement ternary synchronous and asynchronous counters.

The third contribution is the implementation of ternary SRAM cells. Three, dual supply-based ternary Static-Random-Access-Memory (SRAM) cell designs are implemented using CNFETs. Two of the proposed ternary SRAM designs are cycle-operator based and the third design is buffer-based.

The proposed designs of ALU, sequential circuits and SRAM are simulated using HSPICE and show considerable improvement in performance when they compared to their counterparts existing in literature.

Finally, a CNFET-based Ternary Logic Processor (TLP) is designed by integrating all the above designs of the ternary ALU, the ternary sequential logic circuits, and the ternary memory. To start with, an Instruction Set Architecture (ISA) is defined for this TLP that consists of instructions of the Register type, Load-store type, Immediate type, and Branch type. Based on the ISA, the architecture of the CNFET-based TLP is proposed and the transistor level designs of the TLPs' fundamental blocks like the Ternary Instruction Fetch (TIF), Ternary Register File (TRF), Ternary Arithmetic and Logic Unit (TALU) and Ternary Data Memory (TDM) are presented. Simulations are performed for the TLP and the TLPs' individual blocks and the performance parameters like the power consumption, propagation delay, and the number of CNFETs required are calculated. In addition to this, the functionality of the processor is verified using a few of the standard programs.

Contents

List of Tables

List of Figures

Nomenclature

ALU	Arithmetic-Logic-Unit
BU	Branching-Unit
CLA	Carry-Lookahead-Ader
CNFET	Carbon-Nanotube-Field-Effect-Transistor
CNT	Carbon-Nano-Tube
CSA	Conditional-Sum-Adder
FAS	Full-Adder-Subtractor
GCD	Greatest-Common-Divisor
HA	Half-Adder
HAS	Half-Adder-Subtractor
IGU	Immediate-Generate-Unit
ISA	Instruction-Set-Architecture
MIT	Massachusetts-Institute-of-Technology
MVL	Multi-Valued-Logic
NTI	Negative-Ternary-Inverter
PC	Program-Counter

PDP	Power-Delay-Product
PTI	Positive-Ternary-Inverter
RISC	Reduced-Instruction-Set-Computer
RRAM	Resistive-Random-Access-Memory
SNM	Signal-to-Noise-Margin
SRAM	Static-Random-Access-Memory
STI	Standard-Ternary-Inverter
T-SRAM	Ternary-Static-Random-Access-Memory
TALU	Ternary-Arithmetic-Logic-Unit
TCU	Ternary-Control-Unit
TDM	Ternary-Data-Memory
TG	Transmission-Gate
TIF	Ternary-Instruction-Fetch
TIM	Ternary-Instruction-Memory
TLP	Ternary-Logic-Processor
TRF	Ternary-Register-File

Chapter 1

Introduction

1.1 Overview & Motivation

CMOS (Complementary Metal-Oxide-Semiconductor) technology has been the driving force behind the growth of the semiconductor industry for several decades. This technology is used in the manufacturing of integrated circuits (ICs), including microprocessors, memory chips, and other electronic devices. One of the main challenges in CMOS technology is the issues arising due to the continuous scaling of transistors. Scaling refers to the process of reducing the size of transistors and other components on an IC while improving their performance [1]. As transistors are made smaller, they can be packed more densely onto an IC, which can lead to higher performance and lower power consumption. However, as transistors are scaled down, various physical and electrical effects become more pronounced, which can make it more difficult to maintain good performance and reliability. These scaling issues have led to a slowdown in the pace of Moore's Law [2], which is the observation that the number of transistors on an IC double approximately every two years. As the demand for smaller, faster, and more energy-efficient ICs has continued to increase, researchers and engineers have been exploring a variety of post-CMOS technologies that could potentially replace or supplement CMOS technology in the future and overcome the scaling issues. Few of the new device technologies that are being studied, as an alternative to CMOS, are the CNFET (Carbon-Nanotube-Field-Effect-Transistor), the FinFET (fin-Field-Effect-

Transistor) and the QDGFET (Quantum-Dot-Gate-Field-Effect-Transistor).

CNFET devices have been found to be energy efficient [3] when compared to the MOSFETs for building digital logic circuits. The design of a modern RISC-based microprocessor using CNFETs by researchers at MIT [4] has ignited interest in this field of study in recent times. Along with the CNFET technology, various post-binary computing methodologies like quantum computing and Multi-valued logic computing are also being explored. Multi-valued logic uses more than two logic levels for computation. Ternary logic is a special case of MVL that uses three discrete logic levels for computation. A ternary system is better than a binary system in terms of the complexity of interconnects and information density. A ternary system requires a lesser number of bits to represent a number when compared to binary. For example, a 10-digit (N-digit) binary number needs only 6-digits ($log_3(2^N - 1)$) in ternary. These advantages make ternary logic more appealing.

In order to build circuits using ternary logic, transistors with multiple threshold voltages are required. Using CMOS for building ternary circuits would require body biasing of transistors increasing the complexity of the design. In the CNFET, the channel is made up of CNTs. By changing the diameter of these CNTs, transistors with different threshold voltages can be obtained, making CNFET a viable candidate for ternary logic circuit implementation.

The goal of the thesis is to design and implement a ternary logic processor using CNFETs (Carbon Nanotube Field-Effect Transistors) technology. Ternary logic circuits, which operate on three logic levels instead of the traditional binary (two levels), have been extensively studied and implemented using CNFETs in the past decade. Previous research papers [5–17] have explored various aspects of CNFET-based ternary logic circuits, including inverters, logic gates, arithmetic logic units (ALUs), and ternary memory. While some studies have discussed the concept of a ternary logic processor [18, 19], none have presented a transistor-level architecture employing CNFETs for such a processor.

To address this gap, the thesis proposes the utilization of ternary logic computation coupled with CNFET technology to design and implement a CNFET-based ternary logic processor. The thesis introduces various essential ternary circuits that serve as the fundamental building blocks of the processor. The first contribution is a new architecture for a power-efficient ternary ALU (Arithmetic Logic Unit). The ALU is responsible for performing arithmetic and logical operations in the processor. The proposed architecture aims to minimize power consumption while maintaining efficient ternary computation.

Next, the thesis focuses on ternary sequential logic circuits, specifically ternary D-flip-flops. These flip-flops are essential components of the ternary register file, which stores data in the ternary processor. The design of these circuits ensures proper storage and retrieval of ternary values. Furthermore, power-efficient designs for a ternary single-cell SRAM (Static Random Access Memory) are proposed. The SRAM cells are used to construct the instruction and data memory of the processor. The designs aim to minimize power consumption while providing reliable ternary memory storage.

Finally, the thesis culminates in the design of a power-efficient CNFET-based ternary logic processor. This processor integrates all the proposed circuits, including the ternary ALU, ternary D-flip-flops, and ternary single-cell SRAM. The proposed circuits and architecture aim to achieve power efficiency and reliable ternary computation, paving the way for further advancements in ternary logic technology.

1.2 Background

1.2.1 Ternary Logic

Ternary logic, as the name suggests, is three valued logic that uses three logic levels for computation. The three logic levels are represented by 0,1 and 2 respectively. A function $f(X)$ is defined as a ternary logic function mapping $\{0,1,2\}^n$ to $\{0,1,2\}$ where X is given by $X_1, \ldots, X_n$. When X_i,$X_j \epsilon \{0,1,2\}$, the basic operations of ternary logic can be defined as:

$$X_i + X_j = max\{X_i, X_j\} \tag{1.1}$$

$$X_i \cdot X_j = min\{X_i, X_j\} \tag{1.2}$$

$$\overline{X_i} = 2 - X_i \tag{1.3}$$

where equations (1.1), (1.2)and 1.3 indicate OR, AND and NOT operations respectively for ternary logic [5].

Inversion is an important function in Ternary Logic. The general inverter with input x and outputs y0,y1,y2 is denoted as [5]:

$$y0 = C0(x) = \begin{cases} 2, & if\ x = 0 \\ 0, & if\ x \neq 0 \end{cases} \tag{1.4}$$

$$y1 = C1(x) = \overline{x} = 2 - x \tag{1.5}$$

$$y2 = C2(x) = \begin{cases} 2, & if\ x \neq 2 \\ 0, & if\ x = 2 \end{cases} \tag{1.6}$$

Hence, three ternary inverters namely, Negative Ternary Inverter (NTI), Standard Ternary Inverter (STI), and Positive Ternary Inverter (PTI) corresponding to outputs y0,y1 and y2 are defined for ternary logic. Their truth table is as shown in Table 1.1.

Table 1.1: Ternary Inverters [5]

Input x	**NTI** (x)	**STI** (x)	**PTI** (x)
0	2	2	2
1	0	1	2
2	0	0	0

Table 1.2: Logic Symbols

Voltage Level	**Logic Value**
0	0
$Vdd/2$	1
Vdd	2

Table 1.3: Truth Table for TOR and TAND

A	**B**	**A(TOR)B**	**A(TAND)B**
0	0	0	0
0	1	1	0
0	2	2	0
1	0	1	0
1	1	1	1
1	2	2	1
2	0	2	0
2	1	2	1
2	2	2	2

The logic values 0, 1 and 2 correspond to voltages 0, V dd/2 and V dd respectively as shown in Table 1.2. The truth table 1.3 for the basic ternary AND and OR functions is derived from these equations, where the OR of two ternary digits is the maximum of the two digits and the ternary AND is given by the minimum of the two digits respectively. The basic ternary NAND, NOR gates and the ternary inverters STI,PTI,NTI are represented as shown in Figure 1.1.

Transistors with different threshold voltages are required for implementation of ternary logic circuits. CNFETs are ideal for implementing ternary logic circuits since

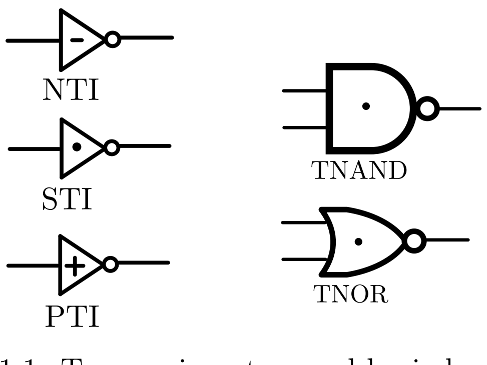


Figure 1.1: Ternary inverters and basic logic gates

using CNFETs of different chiralities different threshold voltages can be obtained. The following section describes the CNFETs in more detail.

1.2.2 Carbon-Nanotube Field Effect Transistor (CNFET)

CNFET consists of CNTs (Carbon-Nano-Tubes) that act as the channel between the two metal electrodes that are the source and the drain instead of the substrate as in the case of MOSFET. A SWCNT (single-walled carbon nanotube) is formed by rolling a sheet of graphite along a certain angle called the chirality vector of the CNT, as shown in Figure 1.2The roll-up vector is represented by $C = na + mb$, where$'a'$ and $'b'$ are unit lattice vectors and m and n are positive integers which specify the chirality of the CNT. Based on the values of the integer pair (n, m) the SWCNTs are further classified into armchair CNT (if $n = m$), zigzag CNT (if $n = 0$ or $m = 0$) and chiral CNT (if $m! = n! = 0$). All armchair CNTs behave as conductors. On the other hand, zigzag and chiral CNTs show metallic (conducting) behavior when $n = m$ or $n–m = 3i$, where i is an integer, otherwise, they show semiconducting behavior. Hence the zigzag and chiral CNTs are used in realizing a CNFET [20].

CNFETs function similarly to MOS transistors in terms of their operating principle. As depicted in Figure 1.3 this three (or four) terminal device is comprised of a semiconducting nanotube serving as a conducting channel and connecting the source and drain contacts. The gate turns the device on or off electrostatically. The drain current is directly proportional to the number of CNTs connected between the source and drain as well as their respective diameters [21, 22].

CNFETs with different chiralities have different threshold voltages. The chirality is also related to the diameter of the CNT. The gate width of CNFET can be approximated using equation below [21]:

$$W \approx \min(W_{min}, N \times S) \tag{1.7}$$

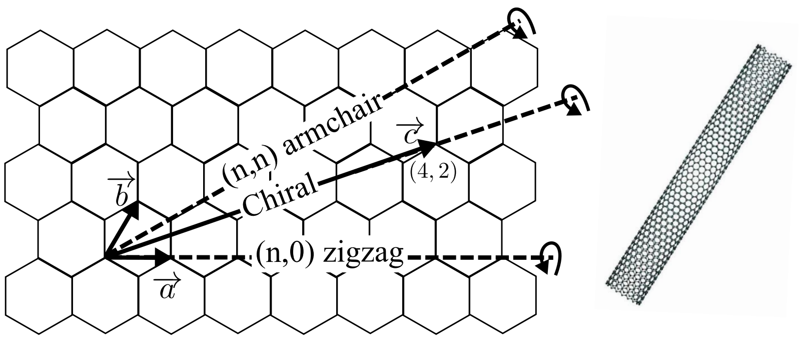


Figure 1.2: Unrolled sheet of graphite and the rolled lattice structure of CNT

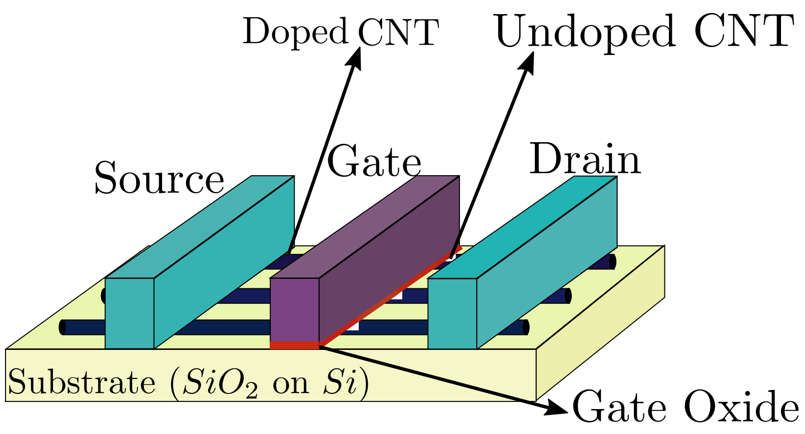


Figure 1.3: Carbon-Nanotube Field Effect Transistor (CNFET) 3-D view

In equation (1.7), W_{min} is the minimum gate width, N is the number of tubes and S the distance between the centers of two adjoining CNTs under the same gate, also called as Pitch. The diameter of CNT, D_{CNT}, which depends on the chirality vector (n, m) can be calculated using equation below:

$$D_{CNT} = \frac{\sqrt{3}a_0}{\pi}(\sqrt{n^2 + m^2 + mn}) \tag{1.8}$$

where $a_0 = 0.142nm$ is the inter atomic distance between each carbon atom and its neighbour. The threshold voltage, which is the voltage needed to turn ON the device electrostatically via the gate, can be approximated to the first order as the half band gap and can be calculated by equation (1.9) [21].

$$V_{th} \approx \frac{E_g}{2e} = \frac{1}{\sqrt{3}}\frac{aV_\pi}{eD_{CNT}} = \frac{0.43}{D_{CNT}(nm)} \tag{1.9}$$

In the above equation, $V_\pi (= 3.033eV)$ is the carbon π - π bond energy in the tight

Table 1.4: Relation Between Chirality, CNT Diameter and Threshold Voltage [21]

Chirality	**Diameter of CNT**	**Threshold Voltage of N-CNFET**	**Threshold Voltage of P-CNFET**
$(19, 0)$	$1.487nm$	$0.289V$	$-0.289V$
$(17, 0)$	$1.330nm$	$0.328V$	$-0.328V$
$(16, 0)$	$1.253nm$	$0.348V$	$-0.348V$
$(14, 0)$	$1.100nm$	$0.398V$	$-0.398V$
$(13, 0)$	$1.018nm$	$0.428V$	$-0.428V$
$(11, 0)$	$0.861nm$	$0.506V$	$-0.506V$
$(10, 0)$	$0.783nm$	$0.559V$	$-0.559V$

bonding model, $a(= 0.249nm)$ is the carbon-carbon atom distance and e is the unit electron charge. If the chirality vector of CNT changes then the threshold voltage of the CNFET will also change. Assuming the m value in the chirality vector is always zero, the ratio of the threshold voltages of two CNFETs with different chirality vectors can be represented by equation below:

$$\frac{V_{th1}}{V_{th2}} = \frac{D_{CNT2}}{D_{CNT1}} = \frac{n_2}{n_1} \tag{1.10}$$

Equation (1.10) shows that threshold voltage of CNFET is inversely proportional to the diameter of CNT which, as mentioned above, depends on its chirality vector. As the CNFET allows for controlling the threshold voltage by changing its CNT diameter, it is suitable for ternary logic circuit implementation.

Table 1.4 illustrates the relationship between chirality, CNT diameter, and threshold voltage as deduced from relationships given in [21]. Well-controlled CNTs are now produced using improved techniques. While it is possible to create CNFETs with any desired chirality, the CNFET-based ternary logic circuits typically uses three chiralities, namely (19,0), (13,0), and (10,0) [5].

In spite of the fact that there are numerous CNFET device models in the literature [21–25], Stanford CNFET device models reported in [26] that are based on work presented in [21, 22] have been widely used for the implementation of CNFET-based circuits. As a result, this work's models use the CNFET model. The Table 1.5lists the technology parameters of CNFET along with a short description and numerical value.

Table 1.5: Technology Parameters for CNFET model in [21,22,26]

Parameter	**Description**	**Value**
L_{ch}	Physical channel length	$32nm$
L_{geff}	Mean free path in the intrinsic CNT channel region	$100nm$
L_{ss}	Length of doped CNT source-side extension region	$32nm$
L_{dd}	Length of doped CNT drain-side extension region	$32nm$
E_{fi}	Fermi level of the doped S/D tube	$0.6eV$
K_{gate}	Dielectric constant of high-k top gate dielectric material	16
T_{ox}	Thickness of high-k top gate dielectric material	$4.0nm$
C_{sub}	Coupling capacitance between the channel region and the substrate	$40pF/m$
V_{fbn} & V_{fbp}	Flat-band voltage for n-CNFET and p-CNFET, respectively	$0eV, 0eV$
$L\_channel$	Physical gate length	$32nm$
$Pitch$	Distance between the centers of two adjacent CNTs	$20nm$
L_{eff}	Mean free path in p+/n+ doped CNT	$15nm$
phi_M	Work function of Source/Drain metal contact	$4.6eV$
phi_S	CNT work function	$4.5eV$

HSPICE simulations are carried out using this CNFET model for ternary logic circuits like ternary ALU, D-flipflops, counters, register, and SRAM cell and performance parameters like power consumption, propagation delay and PDP are calculated. Each of these circuits are then integrated to implement a single-cycle ternary processor in HSPICE which uses this CNFET model. The outline of the work carried out in this thesis is described in the next section.

1.3 Thesis Outline

This thesis is organized as follows:

Chapter 2 presents in brief the literature review of the design of CNFET-based ternary logic circuits. An overview of the various design approaches for ternary logic, the basic and complex ternary combinational logic circuits, ternary sequential logic circuits and ternary memories proposed in literature till date is given. Then, based on the research gaps identified from the literature review, objectives are framed for the thesis.

Chapter 3 describes the design of a ternary ALU implemented using CNFETs. The proposed ternary ALU reduces circuit complexity by eliminating few blocks of the existing ternary ALU architecture and using a 2:1 multiplexer based approach to design the functinal blocks of the TALU. The 2:1 multiplexer based design approach helps in improving the power efficiancy of the overall design. A complete TALU that performs arithmetic operations like addition, subtraction, multiplication, and comparison and logic operations like Ex-OR, OR, AND, NOR and NAND is designed, simulated and is compared with the existing TALU designs for performance.

Ternary sequential circuits using CNFETs are given very meager attention in the literature. Chapter 4 talks about novel design methodologies to implement efficient ternary sequential logic circuits like ternary D-flipflops. Four new designs of ternary D-flipflops are presented. The first design is a ternary buffer and STI based design, the second design is a multiplexer based design that uses sucessor-predecessor circuits for designing D-flipflops. The third and fourth designs are hybrid designs. All the four presented designs show improvement in terms of circuit parameters compared to existing designs.

Memory is an important part of any system. Chapter 5 presents new power-efficient designs of a ternary SRAM cell. Two new cycle operator-based designs and one ternary buffer based design is presented.

The ternary ALU, ternary D-flipflops and ternary memory implemented in the above chapters are used to build the fundamental blocks of the processor which are in-turn used to implement the CNFET-based ternary logic processor, in Chapter 6. An instruction set architecture is proposed for this processor and the working of the processor design is checked by implementing a few programs on it. The functionality of the proposed CNFET-based ternary processor is also verified by implementing it in Simulink.

Finally, Chapter 7 sums up the work, draws conclusions from the key outcomes and lists the main contributions of the research work. The objectives for the future work are also proposed in this chapter.

Chapter 2

Related work

2.1 Literature Review

In CNFET technology, the threshold voltage is regulated by altering the diameter (i.e., Vth dependent on physical dimension) of the CNT, which in turn depends on the chirality vector, as opposed to MOS technology, where body biasing is employed to control threshold voltages. Because of this dependency, CNFETs can be used to build MVL circuits. The recent demonstration of a CNFET-based modern RISC-based (Reduced-Instruction-Set-Computer-based) processor by MIT (Massachusetts Institute of Technology) researchers [4] has rekindled interest in CNFETs, which had been diminishing in recent years due to complex manufacturing technology and reliability concerns. This processor is the conventional binary logic processor designed entirely using CNFETs, that served as the inspiration for this thesis, which endeavours to design a CNFET-based ternary logic processor.

CNFETs have the potential to be useful in computations involving ternary logic. The concept of ternary logic or the use of three logic levels instead of two, dates back to the thesis presented on ternary digital systems in 1967 [27]. It presented the algebra for ternary logic and the design of basic gates. A standard STI gate design, a few logic gates like the NAND and NOR gate and a half adder, were first proposed in [5]. The design of ternary inverters NTI,STI and PTI are as shown in Figure 2.1. These designs are derived from the truth table for ternary inverters. Also, the basic ternary NOR

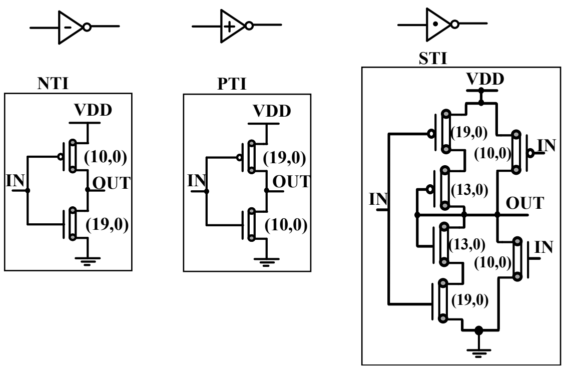


Figure 2.1: Design of ternary Inverters using CNFETs

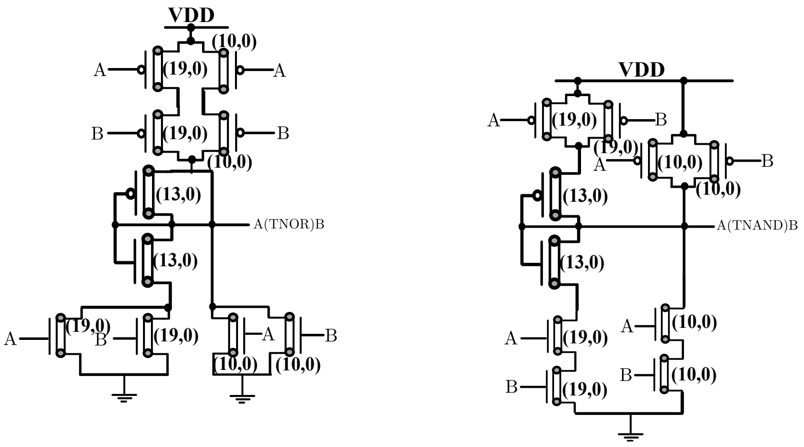


Figure 2.2: Design of ternary NOR and NAND gates

and NAND gates designs are shown in Figure 2.2.

The next sub-sections describe the design of various ternary combinational, sequential and memory circuits that are available in literature so far, which is a step towards the design of a ternary logic processor.

2.1.1 Design of Basic Ternary Combinational Logic circuits

The basic ternary combinational logic circuits like logic gates, STI (Standard Ternary Inverter), half adder, full adder, 1-digit multiplier and comparator are presented using CNFETs in different research papers till date. Various design approaches are used to implement the basic ternary combinational logic circuits. Six different CNFET-based ternary logic circuit design approaches can be found in literature:

1) Decoder-Encoder based approach

2) 3:1 Multiplexer-based approach

3) Decoderless approach

4) Low-power encoder approach

5) 2:1 Multiplexer-based approach

6) Dual supply based approach

Various half adder designs that are implemented using these design approaches are as presented below.

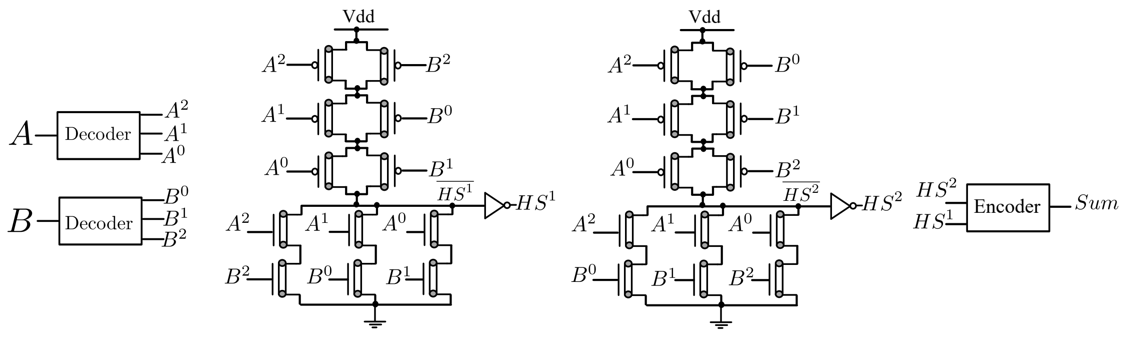


Figure 2.3: Approach I: Decoder-Encoder based HA [5]

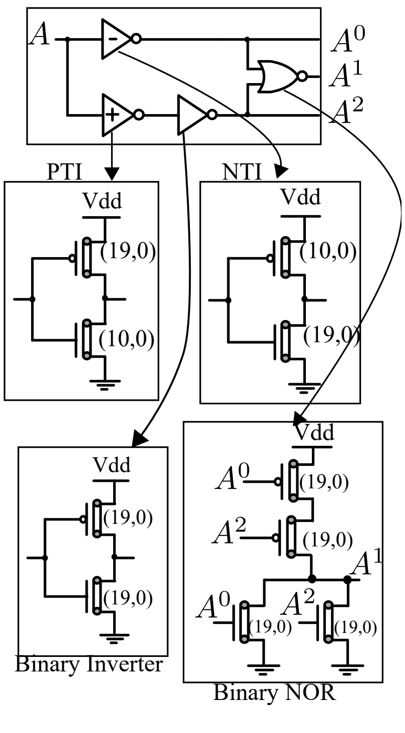


Figure 2.4: Ternary decoder used in Approach-1

The decoder-encoder-based method [5], Approach-I, for designing ternary logic circuits can be divided into three stages. In the initial stage, a ternary decoder converts a ternary signal into mutually exclusive unary functions with two logic levels, logic 0 and logic 2. These decoder outputs can only accept logic values 2 and 0, which correspond to logic 1 and logic 0 in binary logic. In the second stage, the outputs of ternary decoders are combined using binary logic gates. The ternary decoder used in this approach in Figure 2.3 is as shown in Figure 2.4. In the third and final stage, the second stage outputs are combined using an encoder to produce the ternary outputs. A ternary encoder is made up of a level shifter and a ternary OR gate. The Figure 2.3, shows a Half Adder (HA) sum designed using the Decoder-encoder-based design approach. This approach utilises a complex encoder and a ternary decoder for each input, that results in large area and power consumption for higher operand sizes.

Next, the 3:1 Multiplexer approach, Approach II, as described in [28] uses unary operators and 3:1 multiplexers, for implementing ternary logic circuits. A ternary HA designed using this approach is shown in Figure 2.5. This approach has advantages in terms of power consumption when compared to Approach I but suffers from large propagation delay. Another method, Approach-III, described in [29], is a decoderless approach that uses a low power encoder and no decoders. This technique results in ternary circuits with low transistor count and hence optimised in terms of area and power. A ternay HA designed using this method is shown in Figure 2.6.

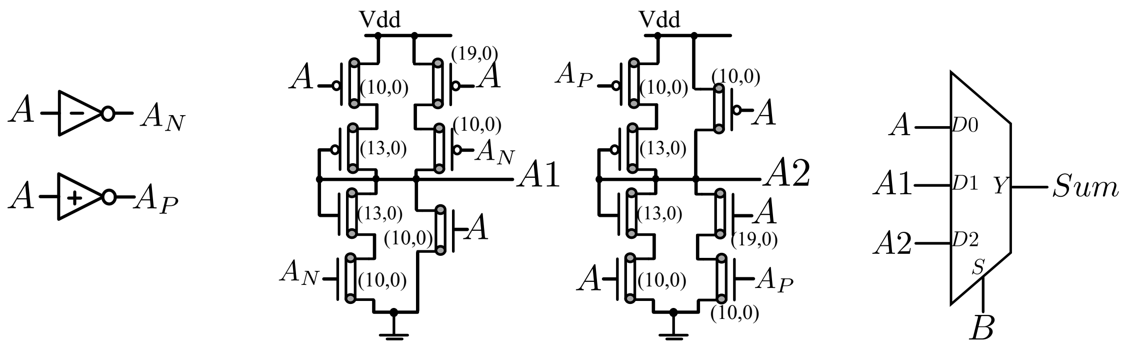


Figure 2.5: Approach II: 3:1 Multiplexer based HA [28]

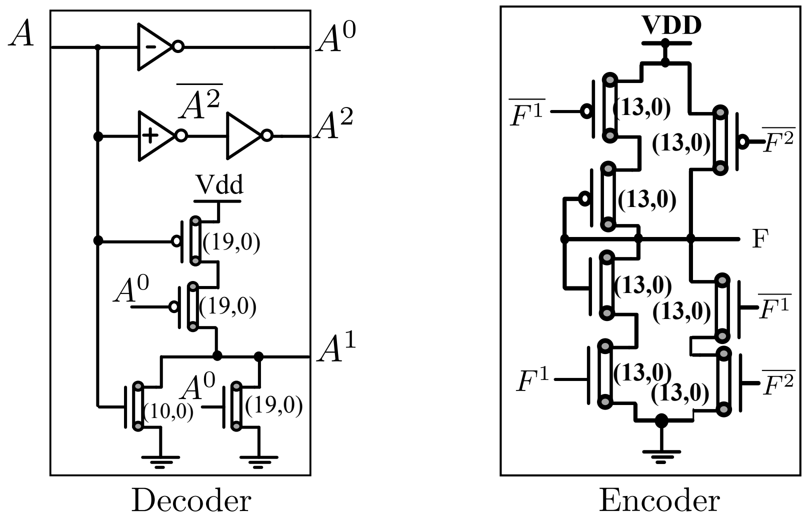


Figure 2.8: Proposed decoder and encoder designs in Approach-III and Approach-IV

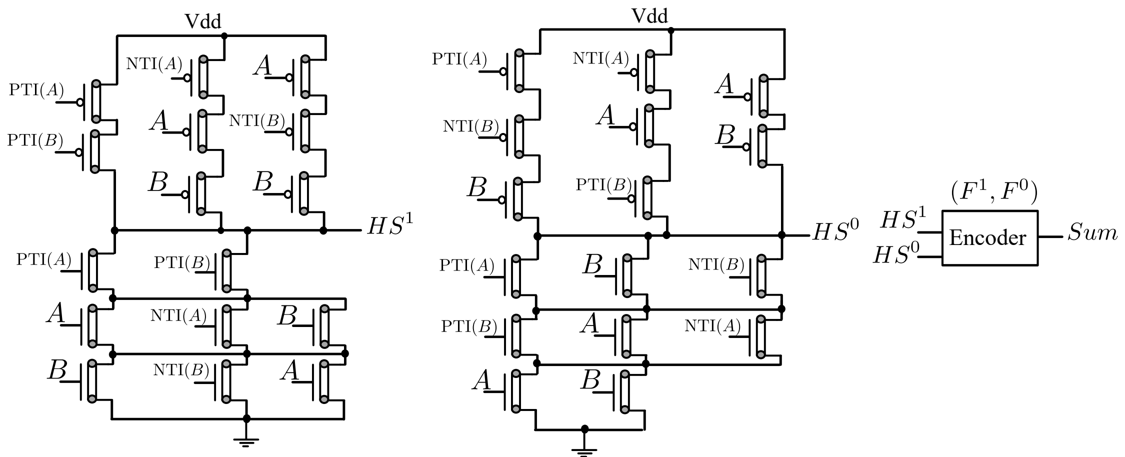


Figure 2.6: Approach III: Decoderless HA [29]

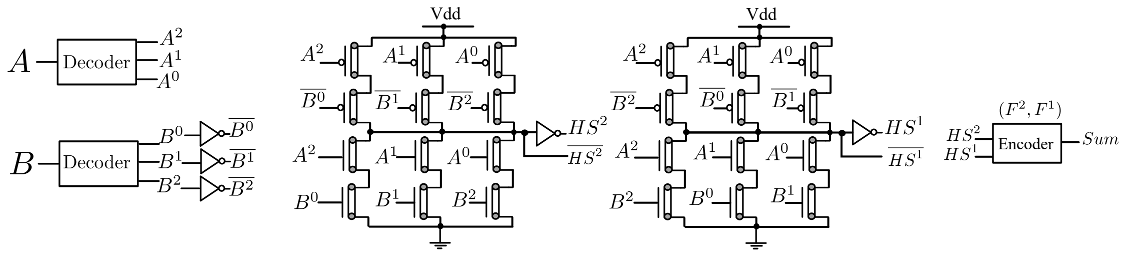


Figure 2.7: Approach IV: Lowdelay decoder based HA [30]

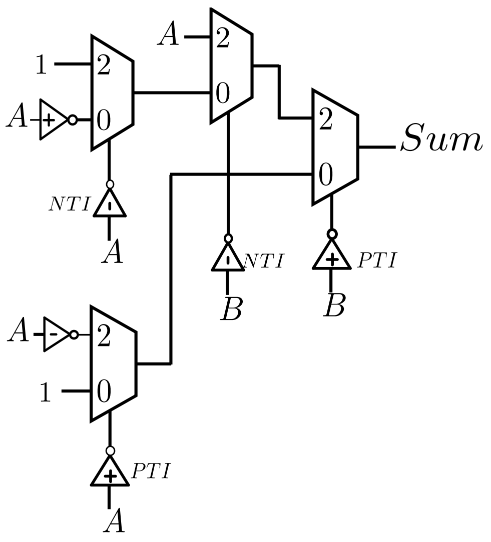


Figure 2.9: Approach V: 2:1 Multiplexer based HA [31]

The Approach-IV, utilises a low delay decoder and the low power encoder to design energy-efficient ternary circuits. The proposed delay optimised decoder is proposed in [30]. The Figure 2.7 shows the HA Sum designed using this approach. Following this, a 2:1 multiplexer based approach, Approach V, was presented in [31], that resulted in ternary circuits which have very low power consumption. The basic idea for the origin of this approach was that the 3:1 multiplexers could be designed using 2:1 multiplexers. Hence, this approach was derived from Approach II. A ternary HA Sum implemented using this design approach is as shown in Figure 2.9.The encoder and decoder proposed in Approach-III and Approach-IV is as shown in Figure 2.8.

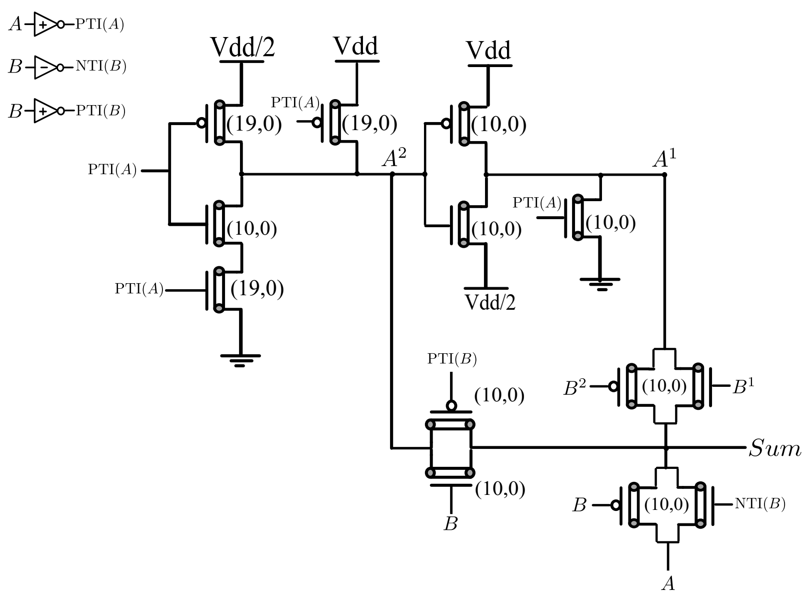


Figure 2.10: Approach VI: Dual supply based HA [32]

Finally, Approach-VI, is proposed recently in [32]. This approach assumes that two power supplies of Vdd and Vdd/2 are present for the implementation of ternary circuits. A HA Sum generation circuit that uses this approach is as shown in Figure 2.10. This approach utlilises the unary operators of ternary system and employs two power supplies, to achieve considerable power consumptions when compared to all other existing design approaches.

Ternary full adder designs using CNTFETs are designed in [10,28,33–35] and multipliers are designed in [36,37]. Some of the recent publications implementing CNFET-based basic ternary combinational circuits are discussed below.

In [38], a novel method for the design of an energy-efficient one-digit adder is proposed. The proposed design utilises ternary multiplexers to select successor and predecessor digits for output node values. This study also defines the novel ternary multiplexer, as well as the successor and predecessor cells. The successor and predecessors here are designed using the VDD/2 based design approach. A ternary full adder modeled using graphene barristors is demonstrated in [39].

The article [40] proposes a CNFET deployed Ternary Half Subtractor and Full Subtractor using unary operators. The design methodology for the implementation of 2-bit ternary comparator utilizing CNFET and resistive random access memory (RRAM) is presented in [41]. The work in [32] proposes new improved designs for ternary half adder and ternary multiplier using the VDD/2 based design approach. In [42], physical equations are used to suggest a systematic way to size transistors for a standard ternary inverter (STI) . This study provides a thorough investigation to establish suitable physical parameter values for the CNFET-based STI.

2.1.2 Design of Complex Ternary Combinational Logic circuits

Multi-digit adders, multi-digit multipliers and ALUs fall under the category of complex combinational logic circuits. Numerous designs of multi-digit adders have been proposed in literarture. Some of these are [8, 10, 28, 33–35]. [35] presents an efficient design of a ternary adder. This adder employs a low-complexity encoder and a fast carry generation device, that results in less propagation delay for multi-digit adders.

Although the encoder used in [35] has reduced delay and complexity, it uses a lot of power.

[8] [30] presents energy-efficient single-digit and multi-digit adders. High-speed and power- efficient designs the ternary prefix adders are presented in [43]. In this paper, a technique is proposed that permits the use of the carry Propagate-Generate concept in multidigit ternary adders. CNFETs are used to implement multidigit ternary prefix adders that use binary prefix trees for payload computation. HSPICE is used to implement five variants of CNFET-based multidigit ternary adders that use distinct prefix networks for payload computation.

In [28], low-delay and low-power single-digit and multi-digit adders have been presented. Although multi-digit CSA (Conditional-Sum-Adder) and CLA (Carry-Lookahead-Ader) designs have low-propagation delays and least PDP, they have complex carry propagation path and consume large power when compared to other designs. In another paper [44], two design approaches for ternary CLA based on K-map and threshold logic methods are proposed in addtion to their realization using CNTFETs only and memristor with CNTFETs. In terms of latency and area, a comparison and tradeoffs among the proposed designs are offered. In the K-map design technique, the comparison demonstrates that the transistor-only implementation is the better choice. However, in the threshold logic (TL) design, the memristor and transistor-based implementation based on memristor and transistor integration is the best.

Few multi-digit ternary multipliers are implemented recently in [45] and [46]. A novel approximate computing technique for low-power ternary multiplication is proposed in [45]. A carry-truncated ternary multiplier, error compensation circuits, and 2×2 ternary multipliers with various accuracies are proposed here using the low-power design methodology with CNFETs. Ternary Wallace tree multipliers that reduce the number of transistors by using 4-input ternary adders are proposed in [46] to improve the performance of existing ternary multipliers. A ternary carry-select adder is also proposed to reduce the carry propagation delay, used as a carry-chain adder of the Wallace tree. The suggested multipliers were made using a custom ternary standard

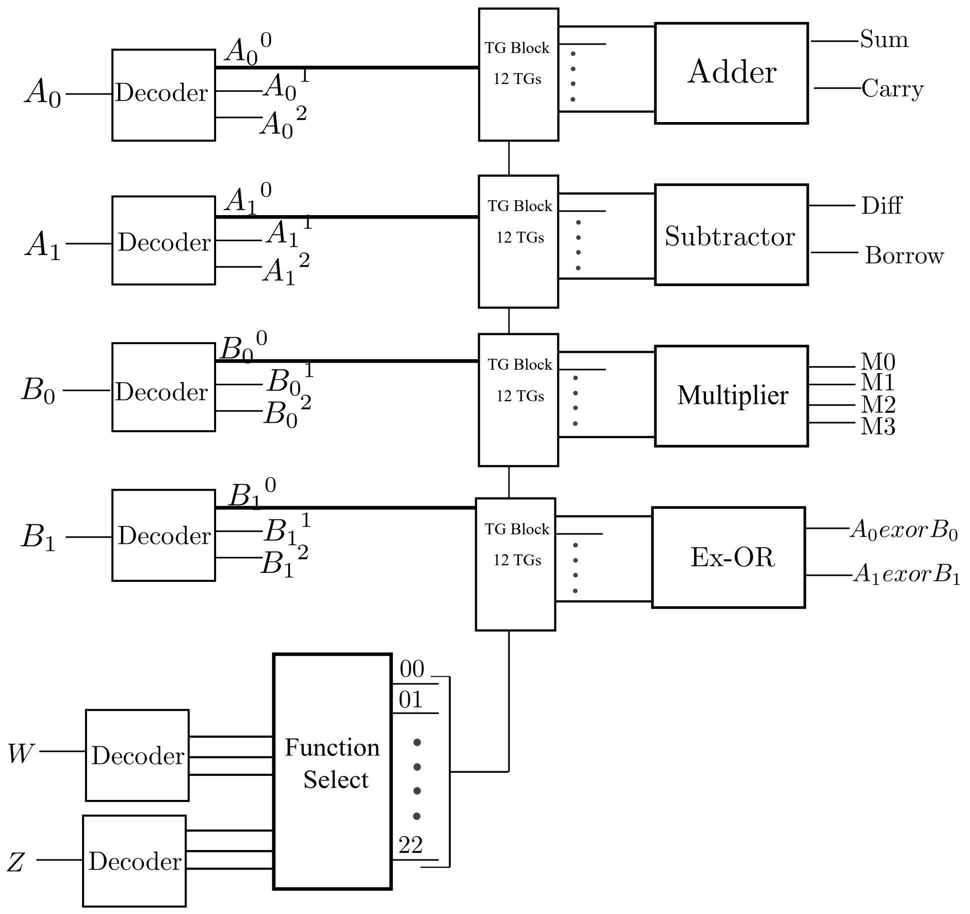


Figure 2.11: Architecture of a ternary ALU first proposed [47]

cell library and a 28-nanometer complementary metal-oxide-semiconductor (CMOS) process.

A 2-digit ternary ALU designed using CMOS technology is presented in [47]. CNFET based Ternary ALUs (TALUs) have been designed in [48] and [11] which have an architecture similar to the one in [47] as shown in Figure 2.11. The basic architecture of the TALU consists of a decoder stage, a function select logic, a transmission gate stage and the functional modules that perform the operations, making the designs complex in terms of the number of transistors required to build the TALU. Also, these circuits use a decoder-encoder based design approach to build their functional modules which consume very high power due to the presence of VDD to Gnd paths in this approach. Another compact and enery efficient TALU design is implemented in [49]. Here, the arithmetic unit, adder and subtractor cells are combined to form a single unit. ALU functional units such as function selection unit and various processing modules are designed using efficient ternary multiplexers rather than the existing transmission gate and switching logic in [11].

2.1.3 Design of Ternary Sequential Logic circuits

The concept of implementing multivalued Sequential logic circuits such as flipflops was first presented in [50]. A ternary edge triggered JKL-flipflop was proposed in [51]. Ternary sequential circuits have been given very less importance in literature so far and hence very few papers are present on this study which gives us a scope for exploring new designs in this area. Another JKL flipflop is presented in [52]. Here, a a novel design of low-power ternary Domino JKL flip—flop on the switch level is proposed. The circuit is simulated by using the Spice tool and the results show that the logic function is correct.

A novel 4−trit pulsed reversible counter is designed with Carbon Nanotube Field Effect Transistors(CNFET) in [53]. In [54], a flip-flop is proposed which captures and propagates a ternary data signal at the four boundaries of a ternary clock signal. It contains four varieties of logic gates: a ternary clock driver, a standard ternary inverter, a binary inverter, and a transmission gate. The results of an HSPICE simulation have confirmed that the power consumption of the flipflop is [55] less than that of conventional single-edge-triggered flip-flops. The work in [56] proposes a negative capacitance CNTFETs(NC-CNTFETs)-based ultra-efficient nonvolatile ternary flip-flop (FF).

Mainly two design approaches are followed in existing CNFET-based ternary D-flipflops, one method uses STIs [58], and the other uses successor-predecessor circuits [57]. The successor circuit implements the next state logic, that is, for an input of logic '0' it outputs a logic '1' and for an input of logic '1' it outputs a logic '2'. The predecessor circuit implements the previous stage logic,that is, for an input of logic $'2'$ it ouputs a logic $'1'$ and for an input of logic $'1'$ it outputs a logic $'0'$. Hence, the successor and predecessor circuits connected back to back act like a storage element and can be used in sequential logic circuits design. The STI-based ternary D-flipflop design is presented in [58] uses a pass transistor based STI gate implementation that

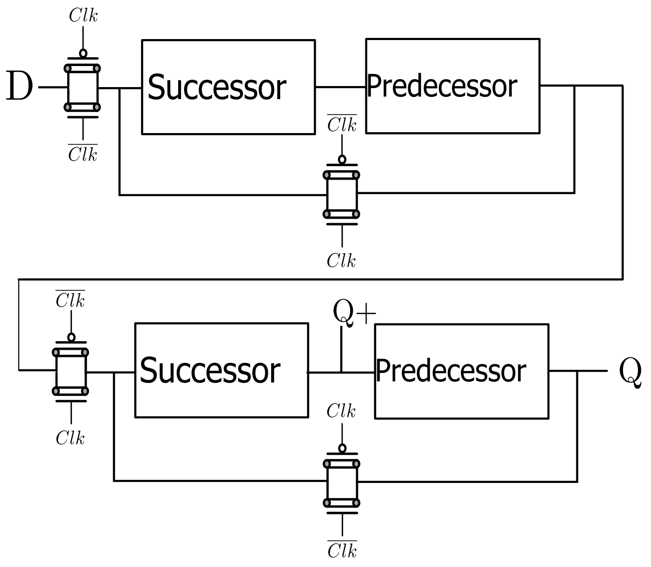


Figure 2.12: Design of a ternary D-flipflop using successor-predecessor circuits [57]

performs better than the standard STI gate in [5] in terms of power as it avoids the VDD to gnd path generation for implementating logic $'1'$ at the output. The designs in [57] and [59] use the successor-predecessor circuits for implementing D-flipflops but these designs are complex in terms of number of CNFETs used and power consumption. Here, the designs for 3-digit ternary synchronous and asynchronous counters are also presented utlising the D-flipflop designs. And since the counter designs are made up of D-flipflops the power consumed by the ternary D-flipfllop also affects the power consumed by the counters. The architecture of a successor-predecessor based ternar D-flipflop is as shown in Figure 2.12.

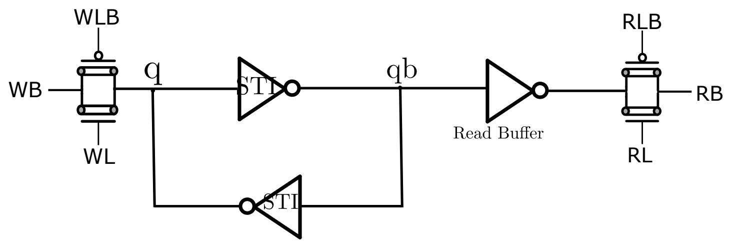
The design methodology for single-edge triggered ternary shift registers is presented in [60]. The D-flip-flops are designed using multiplexer-based positive and negative latches. Following this, series connection of D-flip-flops is done to build serial input serial output and parallel input serial output registers. The parallel input serial output registers are capable of operating in two modes: loading and shifting, and three configurations of AND-OR logic, NAND logic, and latch-based selection circuitry are proposed. According to simulation results, the proposed ternary shift registers reduce power consumption and energy consumption by more than 80% compared to their contemporary counterparts. Recently, a three-valued D-flip-flop (D-FF) circuit and a two-stage shift register built from InGaAs-based multiple-junction surface tunnel transistors (MJSTT) and Si-based metal-oxide-semiconductor field effect transistors (MOSFET) have been demonstrated in [55].

2.1.4 Design of Ternary Memories

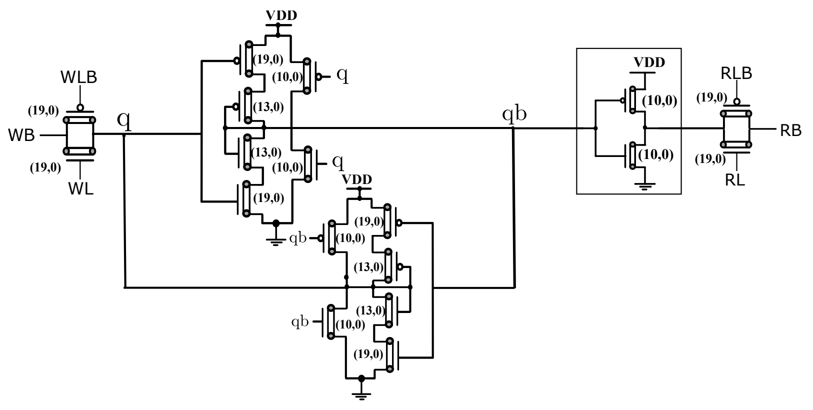
The design of ternary memory using the CMOS technology was presented in [61, 62]. The first ternary memory design using CNFETs was proposed in [17]. In this paper, a new design of a ternary memory cell which uses seperate read and write mechanisms is proposed. Simulation results using HSPICE have showed that the proposed CNTFET-based ternary cell performs the correct function during the read and write operations. It has also been shown that the proposed ternary cells achieve a high SNM due to the separate read and write operations, more than 90 lower standby power consumption for the "0" and "2" states and low area compared to a conventional binary CMOS implementation [17]. Back to back inverters are used in traditional memory cells as basic components of the storage element for the correct states; access transistors (such as pass or transmission gates) are commonly used to read and write from the back to back inverters. The design of the CNFET-based ternary SRAM in [17] is as shown in Figure 2.13.

Another paper in [63] proposed a novel design of a ternary SRAM cell. Unlike previous ternary cells, the proposed cell does not require a read buffer for changing the voltage level of the read bit line, because it uses additional CNTFETs to sink the bit lines to ground. By using four additional CNTFETs for ternary operation, a conventional (two-valued) sense amplifier is then used for output response [63]. The CNTFETs of the ternary cells were sized by grouping them; the chirality of the CNTFETs was modified to improve the SNM and reduce power dissipation while maintaining balanced operation. The chirality of the CNTFETs in the write and read circuits has been enhanced to minimise write/read times in a ternary CNTFET SRAM, producing considerable gains in PDP for the proposed SRAM cell.

The work in [64] proposes two new ternary CNFET-based SRAM cells. The first suggested CNFET SRAM sinks the bit lines to ground using extra CNFETs; its op-



(a) Basic block diagram



(b) Circuit diagram

Figure 2.13: CNFET-based single-cell Ternary SRAM a) basic block diagram b) transistor-level diagram [17]

eration is nearly independent of the ternary values. The second cell incorporates the traditional voltage controller (or supply) of a binary SRAM into a ternary SRAM by adding two CNFETs to the first suggested cell. CNFET characteristics (such as size and density), performance measures (such as SNM and PDP), and write/read timings are all extensively studied and evaluated.

In [65], two ternary SRAMs with reduced delay as compared to their predecessor are proposed. Both proposed SRAMs make use of an enhanced inverter, which is a key component of SRAMs.

[15] demonstrates a read-disturb-free, ternary SRAM cell that uses 17 Carbon Nanotube Field-Effect Transistors. (CNFET). The suggested ternary SRAM cell works on two voltage levels and can store three voltage levels. The proposed SRAM is energy-efficient because the Power Delay Product (PDP) is less during write and read processes than in previous designs.

A STI structure that consumes little current when the input voltage is VDD/2, is presented in [66]. In addition, a ternary SRAM cell was created by employing the storage element with back-to-back STIs and read/write schemes for ternary SRAM cells were developed. To confirm the operation stability of a ternary SRAM cell, three varieties of SNMs (Signal-to-Noise-Margins), were measured. By simulating a pre charging circuit, a ternary SRAM cell, and a write driver, write operation was implemented and the possibility of 'read' operation was demonstrated for a ternary SRAM cell.

A new ternary Static Random Access Memory (T-SRAM) cell is proposed in [67]. Carbon nanotube field-effect transistors are chosen as a proof-of-concept to verify the functionality of the suggested T-SRAM, whereas either post-CMOS or CMOS technologies can replace it. This T-SRAM greatly reduces leakage power and boosts robustness by eliminating the need to store the intermediate ternary state'svoltage level. SPICE simulation and comparison of the proposed T-SRAM with CMOS SRAMs in low-power edge AI applications reveals that it can be a promising substitute.

2.1.5 Design of a Ternary Processor

The first CarbonNanotube computer was proposed in [68], which was built entirely using CNFETs. This was a binary processor that could run MIPS instructions. Then, another CNFET-based modern RISC based microprocessor was built in [4]. Taking inspiration from these demonstrations of CNFET-based binary processors, the concept of a Ternary Logic Processor is described in [18, 19]. The design and verification frameworks for a ternary logic processor are described in [18]. It shows the design and architecture of a RISC based ternary processor that operates on 9-digit long data, called ART-9 (Advanced RISC-based ternary). Although this work describes the block level architecture, the software-level framework and the hardware-level framework of a ternary processor, the complete design of the processor at the transistor level is not proposed. Another paper [19] details a VHDL-based efficient ISA for a ternary logic processor. For a 4-digit ternary processor, the proposed ISA specifies 21 instructions. This paves the way for the development of more efficient ISAs for future ternary logic processor architectures.

2.2 Research Gaps and Objectives of the thesis

The Research gaps were identified in the literature review described in the previous section. Designs for ternary ALU using CNFETs existing in literature consume large power and require a large number of transistors for its implementation. This is due to the complex designs where the decoders are used at the inputs and a decoder-encoder design approach is used to build the functional modules of the ALU. The decoder stage also increases the complexitiy of the transmission gate stage in the ALU. Also, the decoder-encoder based design approach is not power efficient. Hence, there is a need to develop an efficient design of a ternary ALU. The design of CNFET-based ternary sequential logic circuits like D-Flipflops, and counters have received very less attention in the literature. The existing D-flipflop designs are implemented using the successor and predecessor circuits for storing a logic value. These successor-predecessor circuits consume lot of power due to the VDD to Gnd paths present in their design. There is scope to develop new design techniques to implement ternary sequential logic circuits which are power efficient. Memory is a vital part of any system. The Ternary Memory block contributes significantly to the overall power consumed by a Ternary Logic Processor. The existing single-cell ternary SRAM designs are not power efficient. Hence, it is required to develop low-power designs for Ternary SRAM. The design of a CNFET-based Ternary Logic Processor at the transistor level has not been presented in literature yet. This thesis proposes the architecture and design of a 3-digit ternary logic processor built entirely using CNFETs. The functionality of this proposed ternary logic processor is also verified using standard programs. Four objectives are defined for this thesis. The first three objectives lead to the ultimate goal of this thesis, which is to design and implement a Ternary Logic Processor. The objectives are listed below:

1. Design of a 2-digit CNFET-based ternary ALU using the power-efficient 2:1 multiplexer-based design approach.
2. New design methodologies for ternary D-Flipflops and ternary counters.
3. Design of new power-efficient ternary single-cell SRAM.
4. Design of a CNFET-based Ternary Logic Processor.

Chapter 3

Design of CNFET-Based Ternary Arithmetic and Logic Unit

3.1 Introduction

A Ternary Arithmetic and Logic Unit (TALU) is the fundamental component of a ternary processor, as it is responsible for carrying out the mathematical and logical operations required by a computer program. The arithmetic operations that a TALU can perform include Ternary addition, subtraction, and multiplication. The ternary logical operations include TAND, TNAND, TOR, TNOR, TXOR, and various shift operations. The TALU retrieves the data stored in registers within the processor and performs the required operations on them. The TALU then stores the result back in the appropriate register. A computer's ALU is responsible for executing instructions and performing calculations essential to its operation. It is used in conjunction with the control unit to run the program's instructions. Without an ALU, a processor cannot perform arithmetic and logical operations, which are required for a variety of tasks, including mathematical calculations, data processing, and decision-making. Overall, the TALU plays a crucial role in the operation of a Ternary Processor and is a key factor in determining its performance capabilities.

In this chapter, we propose a new design for a TALU that is simpler in terms of area, number of CNFETs used for design, power, and PDP (Power-Delay-Product) metrics as compared to existing TALU designs. The proposed design uses a 2:1 multiplexer based design approach for implementing the arithmetic circuits making the design energy efficient.

3.2 Existing Designs for TALU

A 2-digit ALU implementation for ternary logic using CMOS technology is presented in [47]. The block diagram of the TALU consists of four main components- decoders, function select logic, transmission gates, and function processing modules. Figure 3.1 shows the architecture of a 2-digit TALU as given in [48]. This architecture is similar in all the three existing TALU designs in [47], [48], and [11]. As shown in Figure 3.1, the 2-digit binary inputs A, B, are first converted to binary form using decoders. These decoded inputs are given to the functional modules via a Transmission gate (TG) block. Each TG block is activated, depending on the values of the select lines S_0, S_1, and the inputs are connected to the corresponding functional module. This 2-digit TALU performs four arithmetic and five logic operations.

The TALU architecture shown in Figure 3.1, takes two 2-digit inputs A_1A_0 and B_1B_0. Each digit/bit is represented by, A_i, where A_0 represents 0^{th} bit, A_1 represents 1^{st} bit, and so on, A_i, represents the i^{th} bit of the number A. The 4 input digits (A_1, A_0, B_1, B_0) are decoded to 3 digit outputs each by the input decoders. A ternary decoder, for an input X gives an output X^k, is defined as:

$$X^k = \begin{cases} 2, & ifX = k \\ 0, & ifX \neq k \end{cases} \tag{3.1}$$

So, a 12-digit wide input in the decoded form goes to the arithmetic modules like adder-subtractor, multiplier, comparator and XOR module via a TG block that has 12 TGs (one TG for each of the 12 inputs). The use of decoders and large TG blocks makes this design complex in terms of number of CNFETs required to build the

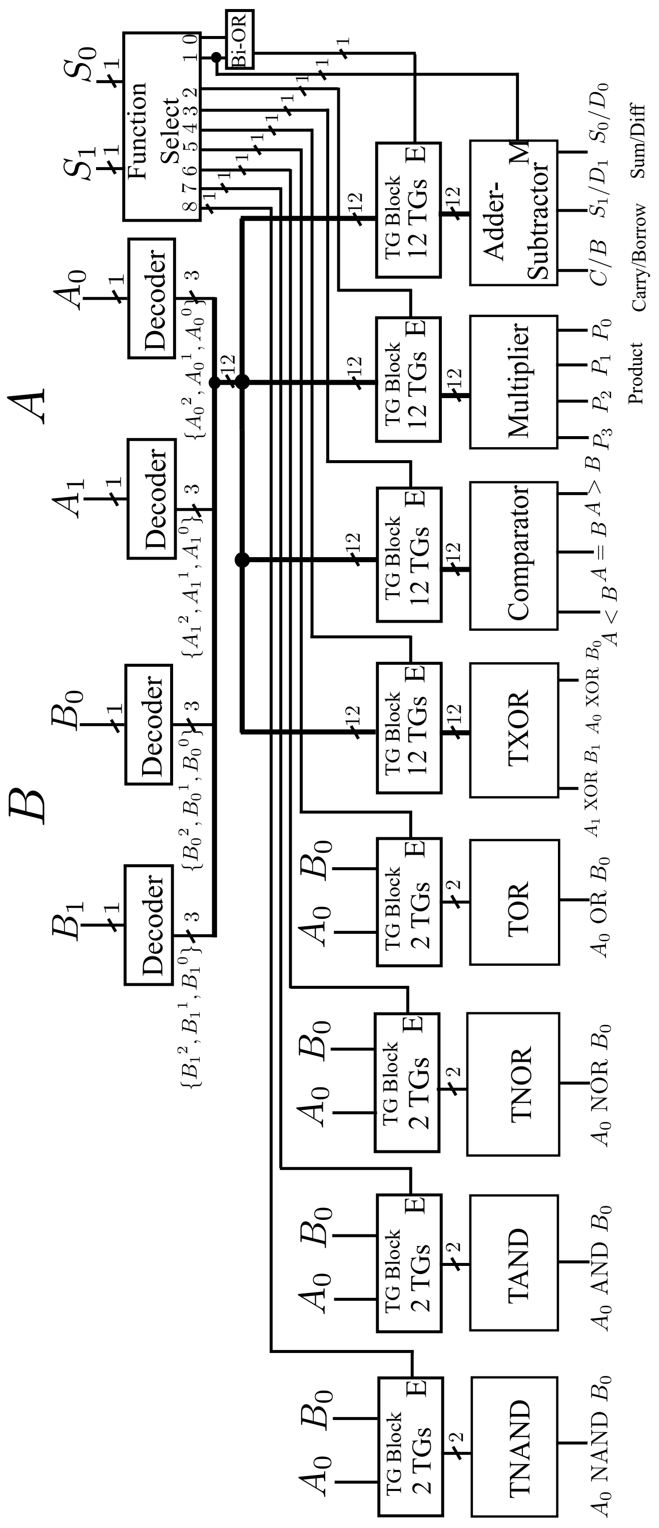


Figure 3.1: Design and Architecture of TALU in [48]

design.The logical modules in Figure 3.1, perform operations on only single digits that require only 2 TGs in the TG block. Each of the functional modules in this design is implemented using a decoder-encoder based design approach which contributes to power consumption.

The architecture of 2-digit TALU in [11] is similar to the one shown in Figure 3.1, except that here it implements the addition, subtraction, and XOR operations in one module hence it is optimized as compared to the design in [48]. Further, in this architecture, additional arithmetic operations like increment, decrement are also performed with the help of a few extra binary gates. Also, a new encoder design is proposed, which is used for implementing the functional modules and yields better performance than the design in [48].

3.3 Proposed Design of TALU

3.3.1 Proposed TALU Architecture

The architecture for a 2-digit TALU proposed in this work, is as shown in Figure 3.2. This TALU performs four arithmetic operations- addition, subtraction, multiplication, and comparison and five logic operations- XOR, OR, AND, NOR and NAND. A digit in binary is called as a bit, so in ternary logic we term it as a trit. The entire TALU block takes two 2-trit numbers as inputs A_1A_0 and B_1B_0 , and gives a 2-trit output TOut(T_1T_0). This output depends on the value of the select lines S_1S_0,which decide the operation that is performed on the inputs. The truth table for the proposed TALU is given in Table 3.1 . The four main components of a proposed TALU design in Figure 3.2, are function select block, transmission gate (TG) block, function processing modules and an output multiplexer module. The function select logic block selects the arithmtic or logic operation that is performed, the TG block transmits the inputs to the processing modules which perform the selected operation and finally the output multiplexer is added to multiplex together the outputs of various operations to a single 2- trit output. Connecting this output multiplexer at the output of TALU design would

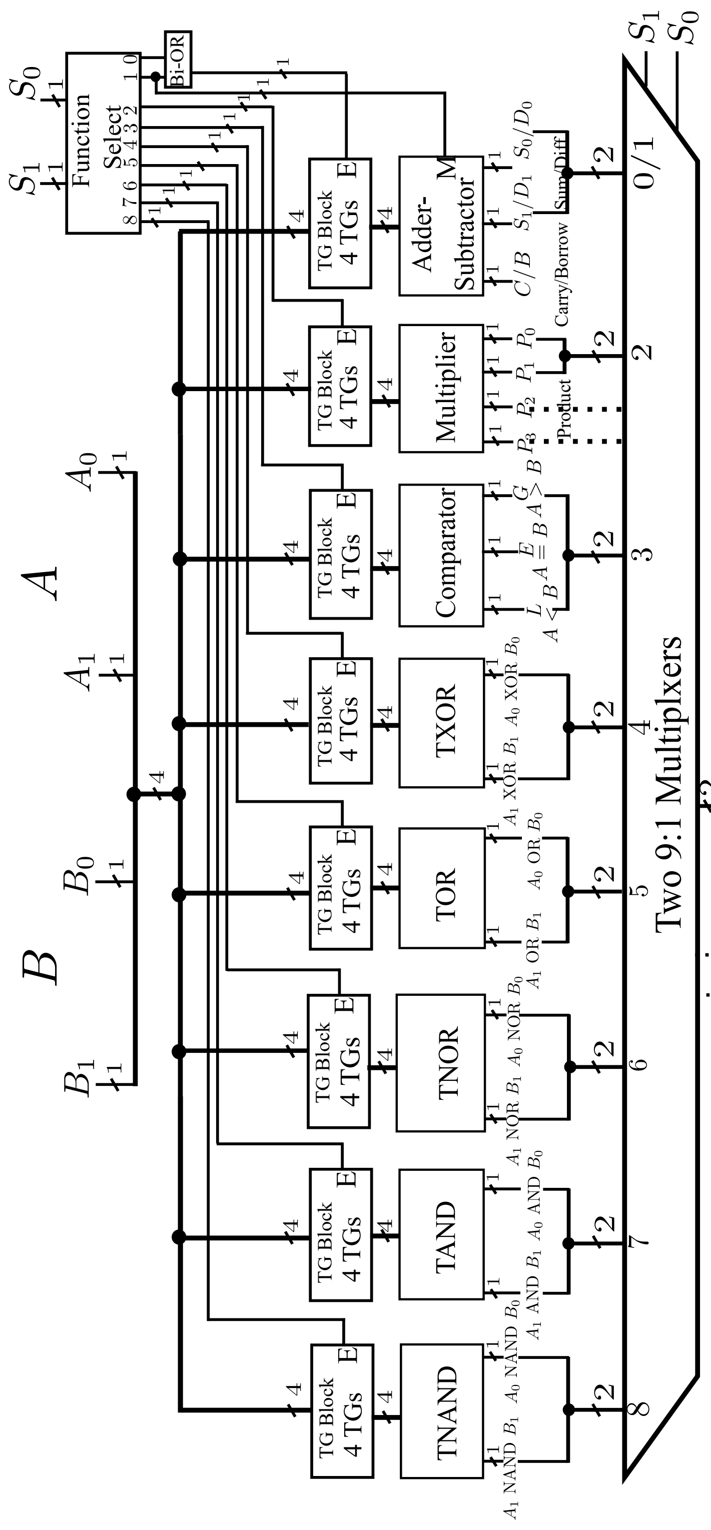


Figure 3.2: Architecture of Proposed TALU

Table 3.1: Truth Table for proposed TALU

S$_0$	**S$_1$**	**Function**
0	0	*Addition*
0	1	*Subtraction*
0	2	*Multiplication*
1	0	*Comparision*
1	1	*EXOR*
1	2	*OR*
2	0	*NOR*
2	1	*AND*
2	2	*NAND*

make the TALU suitable for use in a processor.

Another design of a TALU is also proposed in this work that includes the negation and shift operations. The architecture of this TALU is shown in Figure 3.3 . Negation has been implemented using STI gates and an Inversion block is added to the ternary ALU design. Similarly, a left shift and right shift block that does logical shift is introduced in the TALU design.

3.3.2 Proposed TALU Design and Implementation

The design approach used to implement different blocks of the proposed TALU architecture is presented in this section. The function select logic (FSL) block design is similar to the design presented in [48]. As shown in Figure 3.4,

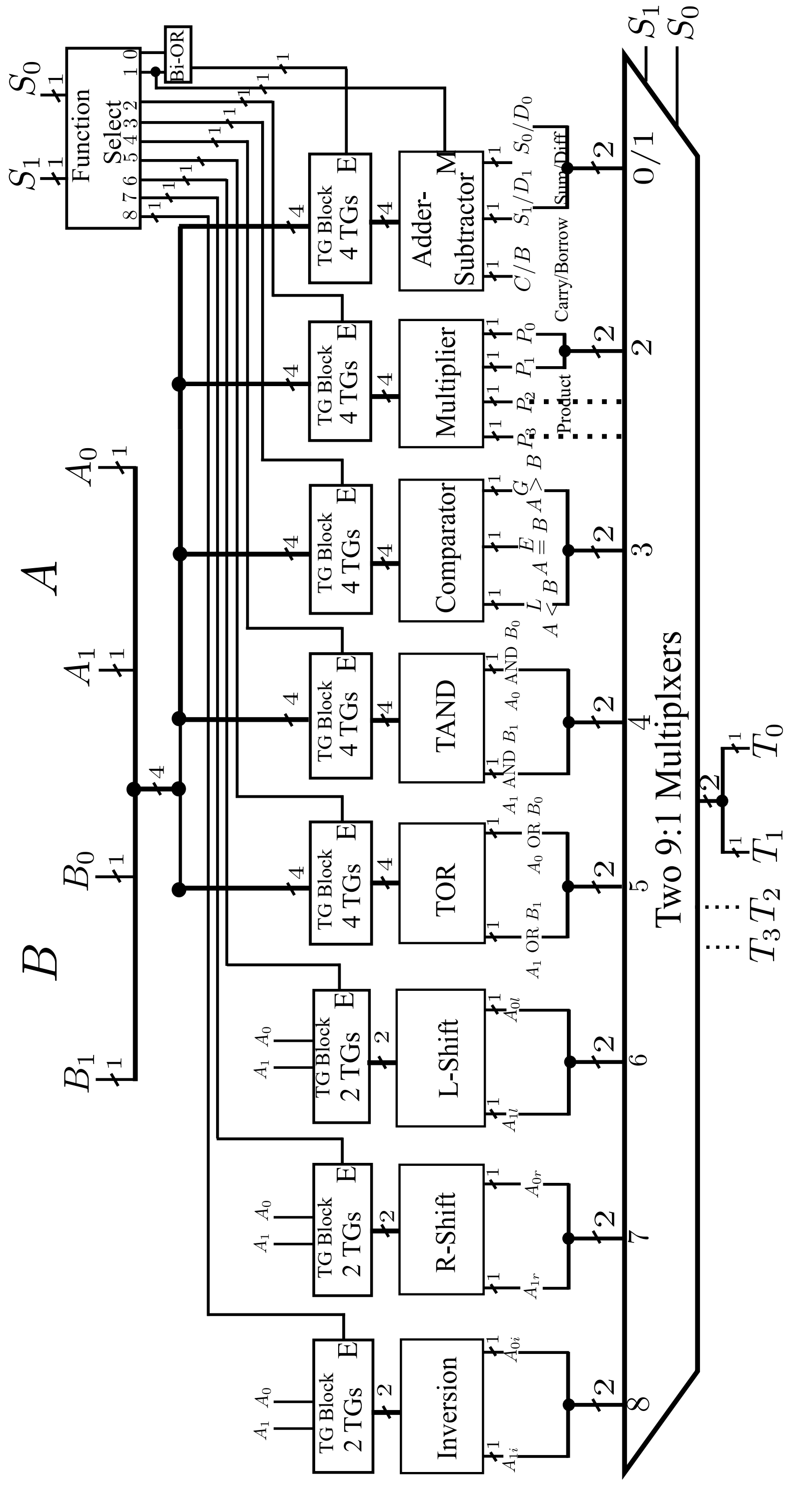


Figure 3.3: Architecture of Proposed ALU with shift and inversion operation

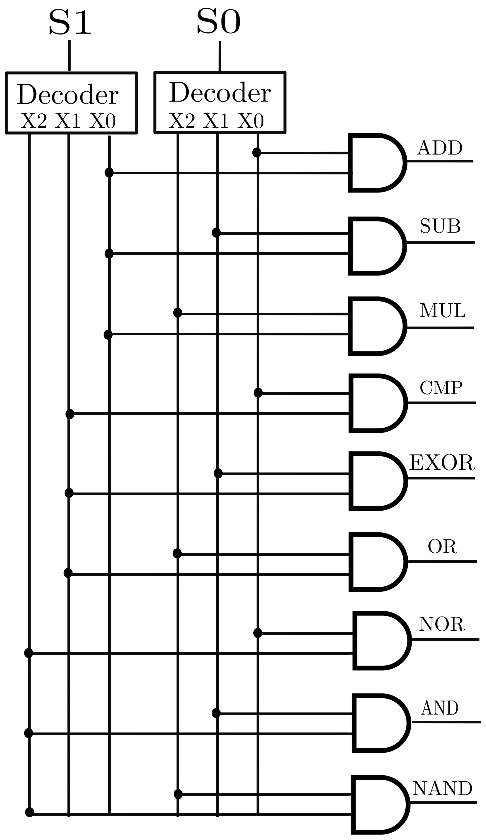


Figure 3.4: Design of Function select logic block [48]

this block has two input select lines S_1S_0 and nine different outputs which act as enable lines. For one particular combination of S_1S_0, one of the nine output lines in enabled and others are disabled. The enabled output in turn enables the TG block which performs the respective function. The FSL block consists of 2 decoders and 9 AND gates. This way nine different functions can be realised.

TG block as the name suggests consists of transmission gates (TGs). This block connects the inputs A_1A_0 and B_1B_0 to the processing modules. We use inputs A and B in their ternary form instead of decoded inputs A and B, resulting in the elimination of input decoders, making the design simpler. Also, here the TALU design takes 4 trits (A_1, A_0, B_1, B_0) as inputs, hence the TG blocks contain only 4 TGs as compared to 12 TGs that are present in the existing TALU designs. In the proposed TALU design as shown in Figure 3.2, all the TG blocks have 4 TGs each. Here, logical operations are performed bitwise for both the input digits, for instance (A_0 OR B_0)and (A_0 OR B_0), which requires 4 TGs to be present in the TG block. If we perform single digit logical

Table 3.2: Unary Operators for Multiplexer based Design

A	A^0	A^2	A^{+1}	A^{+2}	$\overline{A^0}$	$\overline{A^2}$	$1.A^2$	$1.\overline{A^0}$
0	2	0	1	2	0	2	0	0
1	0	0	2	0	2	2	0	1
2	0	2	0	1	2	0	1	1

operations instead 2 TGS would suffice as done in existing TALU design architecture. Due to the elimination of input decoders and TGs, the proposed design uses upto 100 transistors lesser than the existing design.

These ternary inputs via the TG block are given to the processing modules that use a multiplexer based design approach. This makes the processing modules perform better in terms of power consumed when compared to the processing modules in existing TALU designs, which are based on a decoder-encoder based design approach.

The arithmetic modules like the adder-subtractor module, the multiplier module, and the comparator module for 2-trit ternary inputs A, B are designed as described in the sections below. The design of arithmetic circuits using multiplexer based design approach requires some unary operators that need to be implemented. Unary operators are functions applied on a single ternary variable. Unlike in binary logic, many unary operators can be defined in ternary logic. Notations of the unary operators used in this paper are described in Table3.2 . The operators A^0 *and* A^2 are obtained from equation (3), which are the decoder outputs of A and the operators $\overline{A^0}$ *and* $\overline{A^2}$ are inverted values of A^0 *and* A^2. The operators A^{+1} and A^{+2} are called the cyclic operators of A, since they result in the successor and predecessor of A respectively. The operators A^{+1} and A^{+2} are denoted with superscript +1 and +2 because they transform the values of A from $\{0,1,2\} \longrightarrow \{1,2,0\}$ and $\{0,1,2\} \longrightarrow \{2,0,1\}$, which is same as the operations $(A+1)$ and $(A+2)$, respectively. A combination of binary and unary operators results in functions like $1.\overline{A^0}$which is a mapping from $\{0,1,2\} \longrightarrow \{0,1,1\}$. In this paper these unary operators are designed using multiplexers as shown in Figure 3.7.

3.3.2.1 Adder-Subtractor Module

The proposed adder-subtractor module uses a multiplexer based design approach. Whereas the designs in [48] and [11] use the decoder-encoder based design approach. This adder-subtractor block performs both addition and subtraction operations based on the mode input M. The SUB output line is connected to M, so when $M = 0$, it acts as an adder and when M=2 it acts like a subtractor. The block level diagram of the adder subtractor module is shown in Figure 3.5,

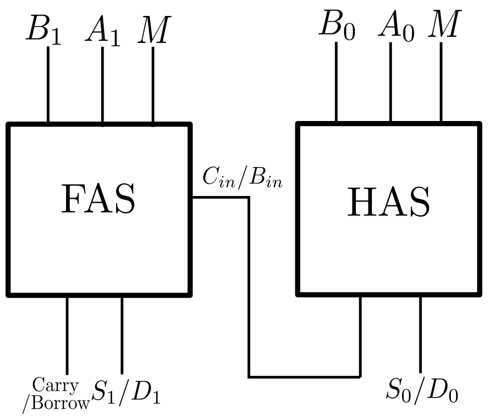


Figure 3.5: Adder-Subtractor block diagram

that takes A_1A_0, B_1B_0, as 2-trit ternary inputs and produces outputs sum S_1S_0, and carry C_1C_0 or difference D_1D_0 and borrow b_1b_0. The operation is performed using a Half Adder-Subtractor (HAS) and a Full Adder-Subtractor (FAS). This module is designed entirely using 2 : 1 multiplexers. The inputs for the HAS are A_0 and B_0 and for the FAS are A_1, B_1 and C_{in} (carry from HAS).

The design uses PTI mux and NTI mux, as described in [69], shown in Figure

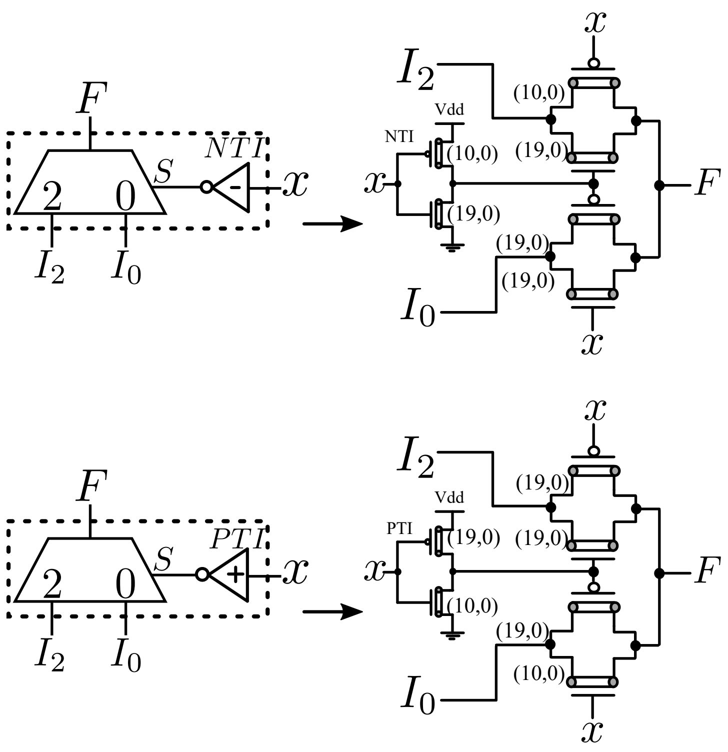


Figure 3.6: NTI Mux and PTI Mux [69]

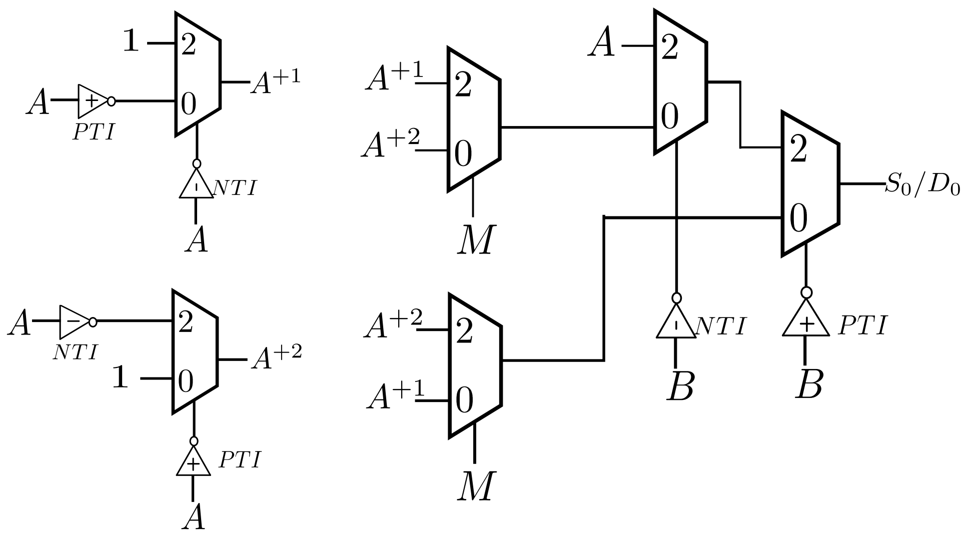
3.6. The multiplexer based design of the HAS module is described in Figure 3.7. The outputs of Sum and carry, obtained from a half adder subtractor truth table 3.3, for $M = 0$ when the module acts as an adder are given as:

Table 3.3: Half Adder-Subtractor Truth Table

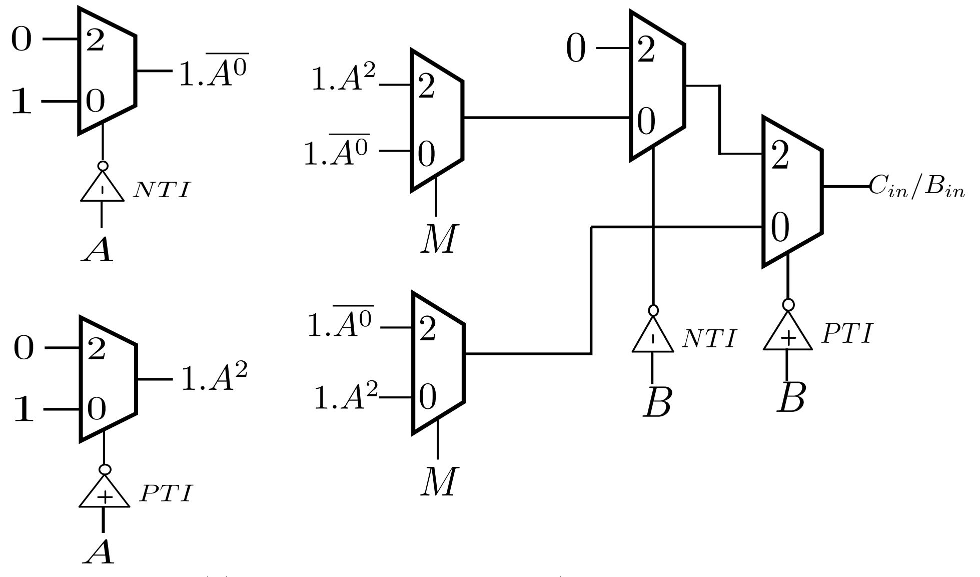
Inputs		**Half Adder**		**Half Subtractor**	
A	**B**	**S**	**C**	**D**	**Bo**
0	0	0	0	0	0
0	1	1	0	2	1
0	2	2	0	1	1
1	0	1	0	1	0
1	1	2	0	0	0
1	2	0	1	2	1
2	0	2	0	2	0
2	1	0	1	1	0
2	2	1	1	0	0

$$SUM : S = B^0.A + B^1.A^{+1} + B^2.A^{+2} \tag{3.2}$$

$$CARRY : C = B^0 + B^1.(1.A^2) + B^2.(1.\overline{A^0}) \tag{3.3}$$



(a) Proposed design for Sum/Difference generation



(b) Proposed design for Carry/Borrow generation

Figure 3.7: Proposed design of Half Adder-Subtractor

where, B is the select line and B^0,B^1,B^2 are related to B, according to the equation (3), and A^{+1},A^{+2},$1.A^2$, $1.\overline{A^0}$ are unary operators as given in Table 3.2. These unary operators are also designed using 2 : 1 multiplexers, with A as select line, as shown in Figure 3.7. These Eqn.s (3.2), (3.3) for Sum and Carry are further transformed as shown below into Eqn.s (3.4),(3.5) which are used for 2 : 1 multiplexer based implementation.

$$S = B^0.A + (B^0 + B^1)(B^1 + B^2).A^{+1} + B^2(B^1 + B^2).A^{+2}$$

$$\because B^1 = (B^0 + B^1)(B^1 + B^2), B^2.B^2 = B^2, B^2.B^1 = 0$$

$$S = B^0.A + (B^1 + B^2)[(B^0 + B^1).A^{+1} + B^2.A^{+2}]$$

$$\because B^1 + B^2 = \overline{B^0} \text{ and } B^0 + B^1 = \overline{B^2}$$

$$S = \underline{B^0.A + \overline{B^0}\left(\underline{\overline{B^2}.A^{+1} + B^2.A^{+2}}\right)} \tag{3.4}$$

$$C = B^0 + (B^0 + B^1)(B^1 + B^2).(1.A^2) + B^2(B^1 + B^2).(1.\overline{A^0})$$

$$C = B^0.0 + (B^1 + B^2)[(B^0 + B^1).(1.A^2) + B^2.(1.\overline{A^0})]$$

$$C = \underline{B^0.0 + \overline{B^0}\left(\underline{\overline{B^2}.(1.A^2) + B^2.(1.\overline{A^0})}\right)} \tag{3.5}$$

where, B is the select line for the PTI-NTI Muxes, as shown in Figure 3.7. The underlined parts in the equations (3.4) and (3.5), are implemented using one 2:1 multiplexer each.

Similarly the ouputs for the half subtractor, for $M = 2$ case, are given as :

$$DIFFERENCE : D = B^0.A + B^1.A^{+2} + B^2.A^{+1} \tag{3.6}$$

$$BORROW : B = B^0.0 + B^1.(1.A^0) + B^2.(1.\overline{A^2}) \tag{3.7}$$

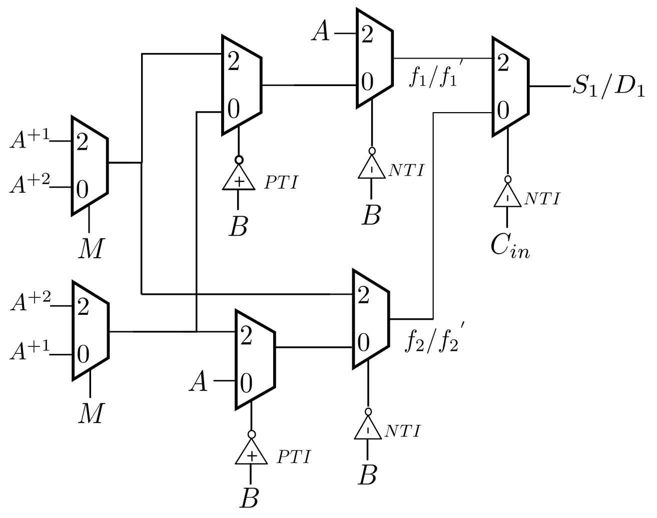
where, B^0,B^1,B^2 are decoder generated unary operators for B, obtained from equation (3), and A^{+1}, A^{+2},$1.A^0$, $1.\overline{A^2}$ are unary operators as given in Table 3.2. These equations for Difference and Borrow are further transformed into Eqn.s (3.8) and (3.9), similar to the Sum Carry equations.

$$D = \underline{B^0.A + \overline{B^0}\left(\underline{\overline{B^2}.A^{+2} + B^2.A^{+1}}\right)} \tag{3.8}$$

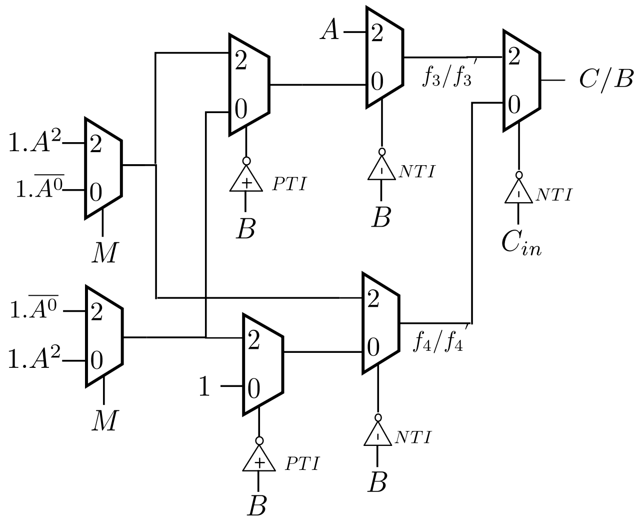
$$B = \underline{B^0.0 + \overline{B^0}\left(\underline{\overline{B^2}.(1.A^0) + B^2.(1.\overline{A^2})}\right)} \tag{3.9}$$

where, B is the select line for the PTI and NTI Muxes, as shown in Figure 3.7.

A Full Adder-Subtractor (FAS) is also designed on similar lines. This design, shown in Figure 3.8



(a) Proposed design for Sum/difference generation



(b) Proposed design for Carry/borrow generation

Figure 3.8: Proposed design of Full Adder-subtractor

, is based on its truth table, as described in Table 3.4 . For ternary inputs A_1, B_1 and C_{in}, this FAS module calculates Sum/Difference and Carry/Borrow. The C_{in} input to the FAS, is the output of HAS block. We represent this input as C_{in} for adder and B_{in}(Borrow in) for subtractor as indicated in the FAS truth table 3.4. From the truth table of HAS, we see that the C_{in} takes values of $0, 1$ and not 2, hence the equations for Sum/Carry contain only C_{in}^0 and C_{in}^1(written as $\overline{C_{in}^0}$) terms. The FAS is entirely designed using $2:1$ multiplexers similar to HAS.

Table 3.4: Full Adder-Subtractor Truth Table

(a) FAS Sum and carry for Cin=0

	Sum			**Carry**		
A/B	**0**	**1**	**2**	**0**	**1**	**2**
0	0	1	2	0	0	0
1	1	2	0	0	0	1
2	2	0	1	0	1	1

(b) FAS Sum and carry for Cin=1

	Sum			**Carry**		
A/B	**0**	**1**	**2**	**0**	**1**	**2**
0	1	2	0	0	0	1
1	2	0	1	0	1	1
2	0	1	2	1	1	1

(c) FAS Difference and Borrow for Bin=0

	Diff			**Bor**		
A/B	**0**	**1**	**2**	**0**	**1**	**2**
0	0	2	1	0	1	1
1	1	0	2	0	0	1
2	2	1	0	0	0	0

(d) FAS Difference and Borrow for Bin=1

	Diff			**Bor**		
A/B	**0**	**1**	**2**	**0**	**1**	**2**
0	2	1	0	1	1	1
1	0	2	1	0	1	2
2	1	0	2	0	0	1

The outputs for Sum/carry for the FAS module, when $M = 0$ and it acts like a full adder are given as:

$$SUM : S = C_{in}^{0}.f_1 + \overline{C_{in}^{0}}.f_2 \tag{3.10}$$

where, $f_1 = B^0.A+ B^1.A^{+1} + B^2.A^{+2}$

$$f_1 = \underline{B^0.A + \overline{B^0}\left(\underline{\overline{B^2}.A^{+1} + B^2.A^{+2}}\right)} \tag{3.11}$$

and $f_2 = B^0.A^{+1}+ B^1.\ A^{+2} + B^2.A$

$$f_2 = \underline{B^0.A^{+1} + \overline{B^0}\left(\underline{\overline{B^2}.A^{+2} + B^2.A}\right)} \tag{3.12}$$

$$CARRY : C = \underline{Cin^0.f_3 + \overline{C^0_{in}}.f_4} \tag{3.13}$$

where, $f_3 = B^0.0+ B^1.(1.A^2) + B^2.(1.\overline{A^0})$

$$f_3 = \underline{B^0.0 + \overline{B^0}\left(\underline{\overline{B^2}.(1.A^2) + B^2.(1.\overline{A^0})}\right)} \tag{3.14}$$

and $f_4 = B^0.(1.A^2) + B^1.(1.\overline{A^0}) + B^2.1$

$$f_4 = \underline{B^0.(1.A^2) + \overline{B^0}\left(\underline{\overline{B^2}.(1.\overline{A^0}) + B^2.1}\right)} \tag{3.15}$$

Similarly the ouputs for the full subtractor, for $M = 2$ case, are given as:

$$DIFFERENCE : S = Cin_0.f_1^{'} + \overline{C^0_{in}}.f_2^{'} \tag{3.16}$$

where, $f_1^{'} = B^0.A + B^1.\ A^{+2} + B^2.A^{+1}$

$$f_1^{'} = \underline{B^0.A + \overline{B^0}\left(\underline{\overline{B^2}.A^{+2} + B^2.A^{+1}}\right)} \tag{3.17}$$

and $f_2^{'} = B^0.A^{+2} + B^1.A^{+1} + B^2.A$

$$f_2^{'} = \underline{B^0.A^{+2} + \overline{B^0}\left(\underline{\overline{B^2}.A^{+1} + B^2.A}\right)} \tag{3.18}$$

$$BORROW : C = \underline{Cin_0.f_3^{'} + \overline{C^0_{in}}.f_4^{'}} \tag{3.19}$$

where, $f_3^{'} = B^0.0 + B^1.(1.A^0) + B^2.(1.\overline{A^2})$

$$f_3^{'} = \underline{B^0.0 + \overline{B^0}\left(\underline{\overline{B^2}.(1.A^0) + B^2.(1.\overline{A^2})}\right)} \tag{3.20}$$

and $f_4^{'} = B^0.(1.A^0) + B^1.(1.\overline{A^2}) + B^2.1$

$$f_4^{'} = \underline{B^0.(1.A^0) + \overline{B^0}\left(\underline{\overline{B^2}.(1.\overline{A^2}) + B^2.1}\right)} \tag{3.21}$$

where, B is the select line and B^0,B^1,B^2 are related to B, according to the equation (3), and A^{+1}, A^{+2},$1.A^2$, $1.\overline{A^2}$, $1.A^0$,$1.\overline{A^0}$ are unary operators as given in Table 3.2.

3.3.2.2 Multiplier Module

A multiplier module performs multiplication operation on two inputternary numbers and generates product and carry as outputs. The 2-trit multiplier module, whose architectue is as given in [11], requires 1-trit multiplier, Half-adder, full adder and carry adders circuits. The half adder and full adders are designed using 2 : 1 multiplexers as described in previous section. The carry adder is designed using the design in [11] and the 1-trit multiplier is designed using 2 : 1 multiplexers. The design for the 1-trit multiplier is as shown in Figure 3.9,

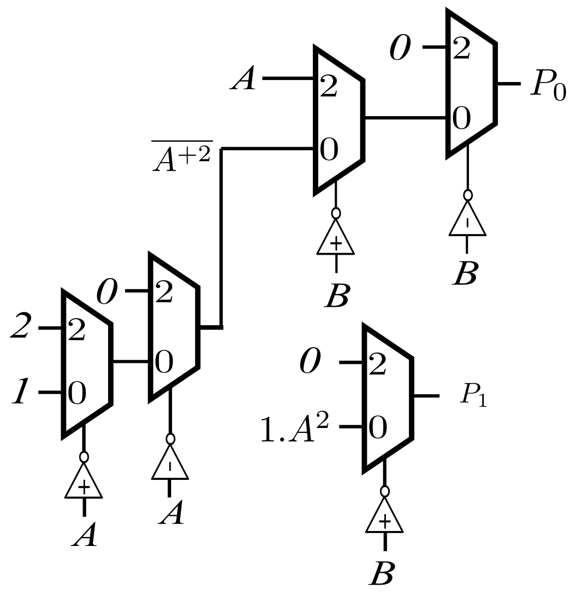


Figure 3.9: Product-Multiplexer based realization

is similar to the design in [70]. The outputs of product and carry for 1-trit ternary inputs, where B is the select line for multiplexer, are given as:

$$PRODUCT : P = B^0.0 + B^1.\left(A\right) + B^2.\left(\overline{A^{+2}}\right) \tag{3.22}$$

$$CARRY\ : C = B^0.0\ + B^2.\left(1.A^2\right)$$

These equations in their 2 : 1 multiplexer based design format, with B as select line, as shown in Figure 3.9 are converted to:

$$\underline{P = B^0.0\ + \overline{B^0}.\underline{\underline{\left(\overline{B^2}.A + B^2.\overline{A^{+2}}\right)}}} \tag{3.23}$$

$$C = \underline{B^0.0\ + B^2.\left(1.A^2\right)} \tag{3.24}$$

where, B is the select line and B^0,B^1,B^2 are related to B, according to the equation (3), and A^{+2},$1.A^2$ are unary operators as given in Table 3.2.

3.3.2.3 Comparator Module

A comparator module compares the two 2-trit input values A_1A_0 and B_1B_0, and gives a 2-trit output *Comp* which is a $02, 20$ for $A < B$ and $A > B$, respectively.The equations for G(Greater) and L(Lesser), in terms of g_0, g_1 and l_0, l_1 where g_0, g_1, l_0, l_1 are greater and lesser signals when digits $A0, B0$ and $A1, B1$ are compared respectively, are given as:

$$G = g_1 + g_0\overline{l_1} \tag{3.25}$$

$$L = l_1 + \overline{g_1}l_0 \tag{3.26}$$

The outputs for signals g and l when 1-trit digits (either MSBs or LSBs) ,are compared , in terms of 2:1 multiplexers(underlined), are given as:

$$g = \underline{B^0.A^0 + \overline{B^0}.\left(\underline{\overline{B^2}.A^2 + B^2.0}\right)} \tag{3.27}$$

$$l = \underline{B^2.\overline{A^2} + \overline{B_2}.\left(\underline{\overline{B^2}.\overline{A^0} + B^0.0}\right)} \tag{3.28}$$

where, B is the select line for the multiplexers as shown in Figure 3.10

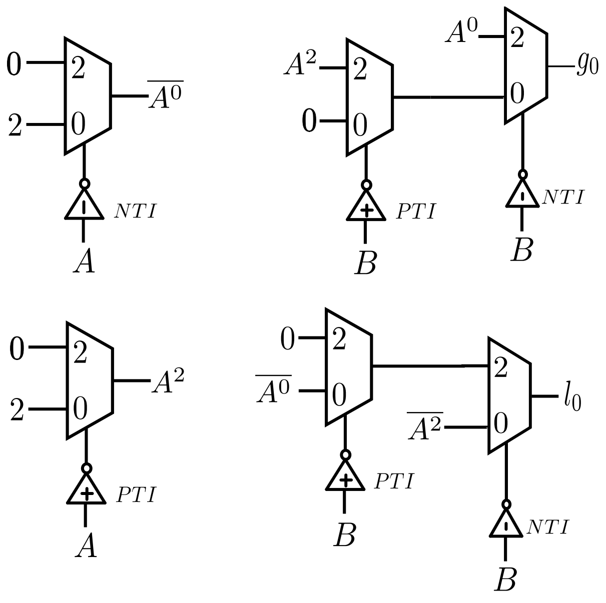


Figure 3.10: Design for 2-digit Comparator

. The unary operators A^0,A^2, $\overline{A^0}$,$\overline{A^2}$ are as described in Table 3.2, are also designed using 2 : 1 multiplexers with A as select line as shown in Figure 3.10.

The signals G and L can be obtained from the signals g_0, g_1, l_0, l_1as given in equation (27) and (28) with the help of two more 2 : 1 multiplexers.

3.3.2.4 Output Multiplexer module

This module multiplexes the outputs of various arithmetic and logic functions into one single TALU output. Figure 3.11 shows the design of the output multiplexer which is a 9 : 1 multiplexer. A 9 : 1 multiplexer needs to be used since there are two ternary select lines S_0 and S_1 using which we implement nine different functions as given in the TALU truth table as shown in 3.1. This 9 : 1 multiplexer is in turn implemented using

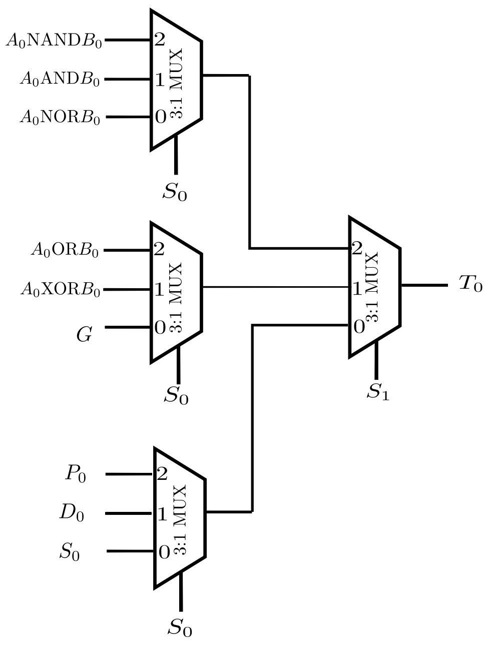


Figure 3.11: Proposed Output Multiplexer design for TALU

four 3 : 1 multiplexers as shown. At the first level, three 3 : 1 mutiplexers are used which has S_0 as the select line and at the second level there is one 3 : 1 multiplexer which has S_1 as its select lines. This way we need two 9 : 1 multiplexers in the output multiplexer module which multiplexes the nine different 2-trit outputs obtained from function modules of the proposed TALU design into a 2-trit TALU Output (T_1T_0).

The outputs from all the functional modules like adder, subtractor, comparator, etc are 2-trit values which are given to the output multiplexer to obtain the final TALU output, except for the Multiplier output. Since it is a 2-trit Multiplier, its outputs are 4-trit values ($P_0 - P_3$), two of which P_0 and P_1 are given to output multiplexer, and the remaining two product output digits P_2, P_3 are given to the output via two TGs that are enabled by same enable line from function select block that enables the multiplier module ($S_0 = 0, S_1 = 2$). Also, all the carry outputs from the functional modules are handled in a similar manner using TGs.

3.4 Simulation Results

3.4.1 Simulation Environment

Simulations were performed using the Standard CNFET model given in [21, 22, 26], in HSPICE, a circuit simulation tool by Synopsys. The Standard CNFET model in [21] is a circuit-compatible compact model for the intrinsic channel region of the MOSFET-like single-walled CNFETs.

The circuits were simulated at room temperature, at 0.9 Volts power supply. CNFETs used in this design are configured to have three tubes, a channel length of 32 nm and a pitch value of 20 nm. For comparison of simulation results of proposed designs with the existing designs, all the designs were given the same randomized input test patterns at a switching frequency of 500 MHz. For delay calculations, the worst-case delay was considered.

3.4.2 Results and Discussion

A new design for a TALU is proposed in this paper, which is implemented using HSPICE. The simulation results obtained are as shown in Table 3.5, 3.6, 3.7, 3.8 and Table 3.9. The TALU designs presented in [48] and [11] are represented as TALU-1 and TALU-2, respectively. And the TALU design proposed in this paper is represented as PTALU. The waveforms obtained for the proposed design PTALU, where $A(A_1A_0)$ and $B(B_1B_0)$ are the 2-trit inputs and TALU Out (T_1T_0) is the 2-trit output, are as shown in Figure 3.12 . For different values of S_0 and S_1, different operations are performed on inputs A and B according to the TALU Truth Table. The waveforms for individual blocks like the Half adder, the Full adder, the 1-trit multiplier, logic gates and shift operations are also shown in Figures 3.13 , 3.14 and 3.15 .

Table 3.5: Simulation results based Comparision of HAS

Designs	**Delay(ps)**	**Power (uW)**	**PDP(x10^{-18}J)**	**CNFETs count**
TALU-1 [48]	35.68	1.72	60.52	214
TALU-2 [11]	14.9	6.25	93.12	54
[71]	35.60	1.71	60.87	210
PTALU	21.2	0.24	5.08	32
Improvement in PTALU w.r.t. TALU-1	40.4%	85.8%	91.6%	85%
Improvement in PTALU w.r.t. TALU-2	−29.7%	96.1%	94.5%	40.7%
Improvement in PTALU w.r.t [71]	40.4%	85.9%	91.6%	84.7%

Table 3.6: Simulation results based Comparision of FAS

Designs	**Delay(ps)**	**Power (uW)**	**PDP(x10^{-18}J)**	**CNFETs count**
TALU-1 [48]	31.17	9.24	286.1	342
TALU-2 [11]	33.7	7.6	256.1	112
[71]	31.19	9.54	297.5	338
PTALU	38.0	0.31	11.7	44
Improvement in PTALU w.r.t. TALU-1	−17.9%	96.6%	95.9%	87.1%
Improvement in PTALU w.r.t. TALU-2	−11.3%	95.9%	95.4%	60.7%
Improvement in PTALU w.r.t [71]	−17.9%	96.7%	96%	86.9%

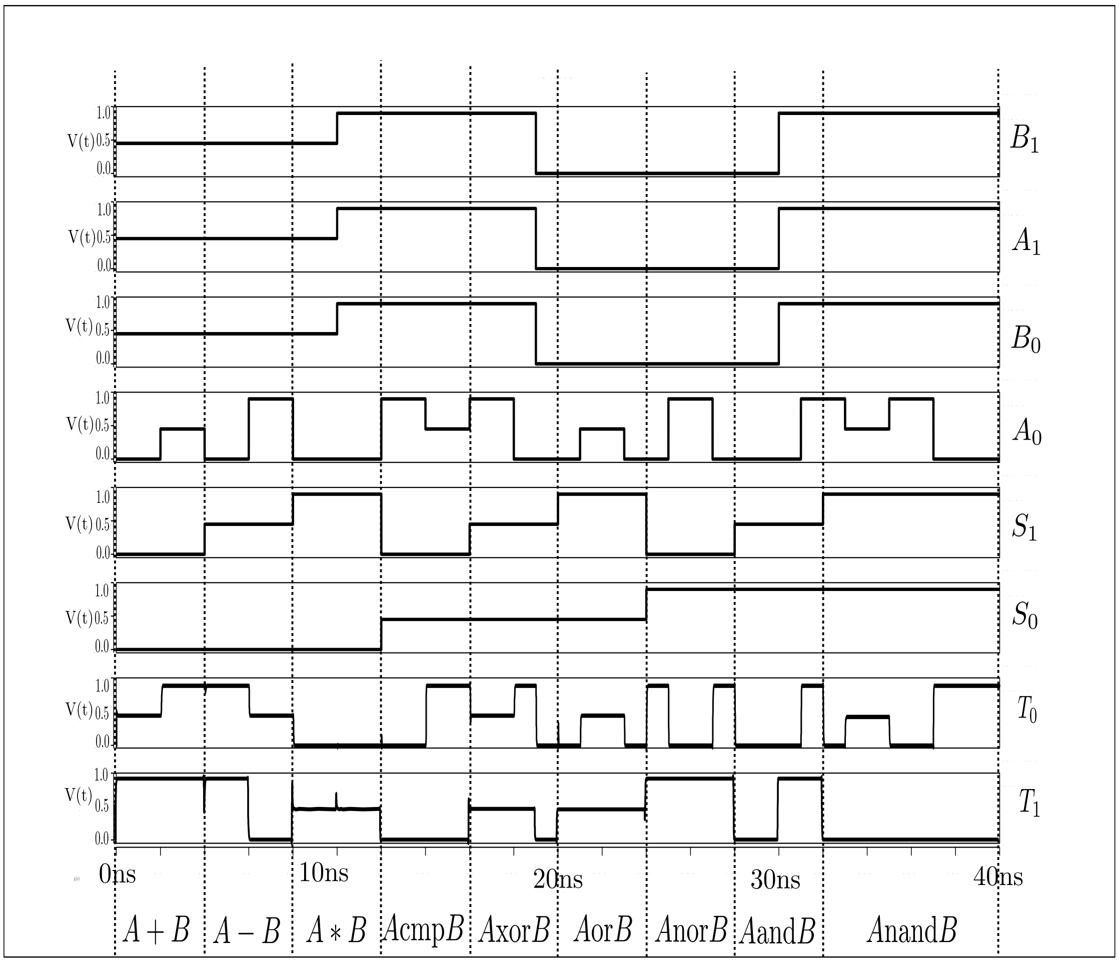
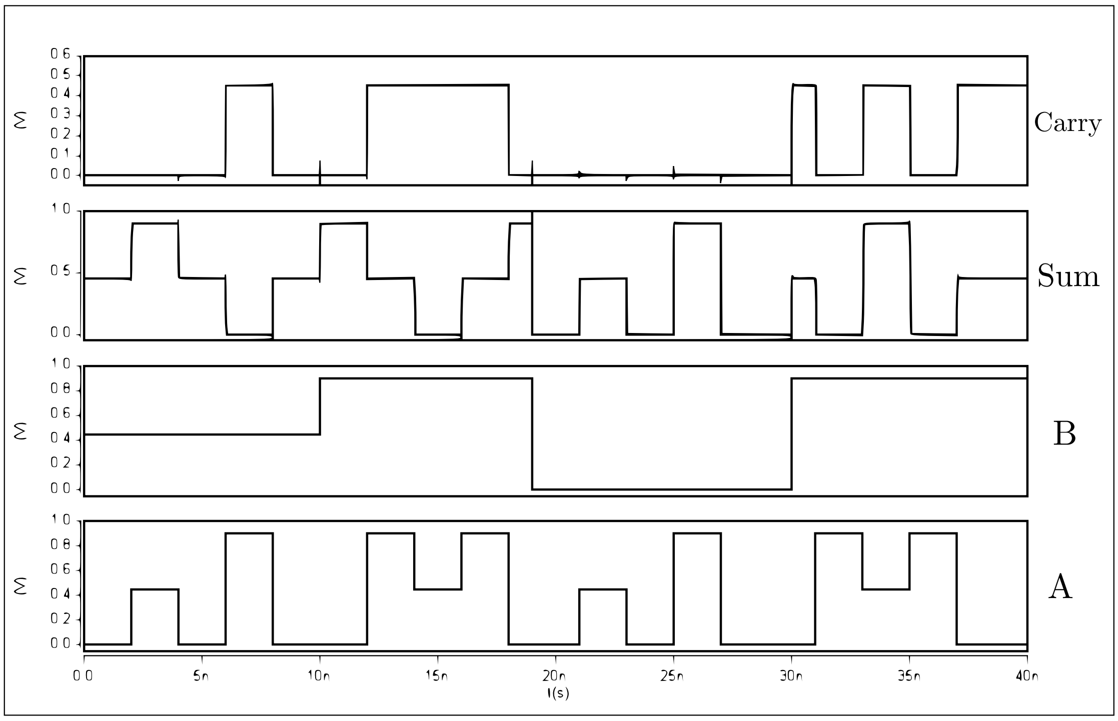
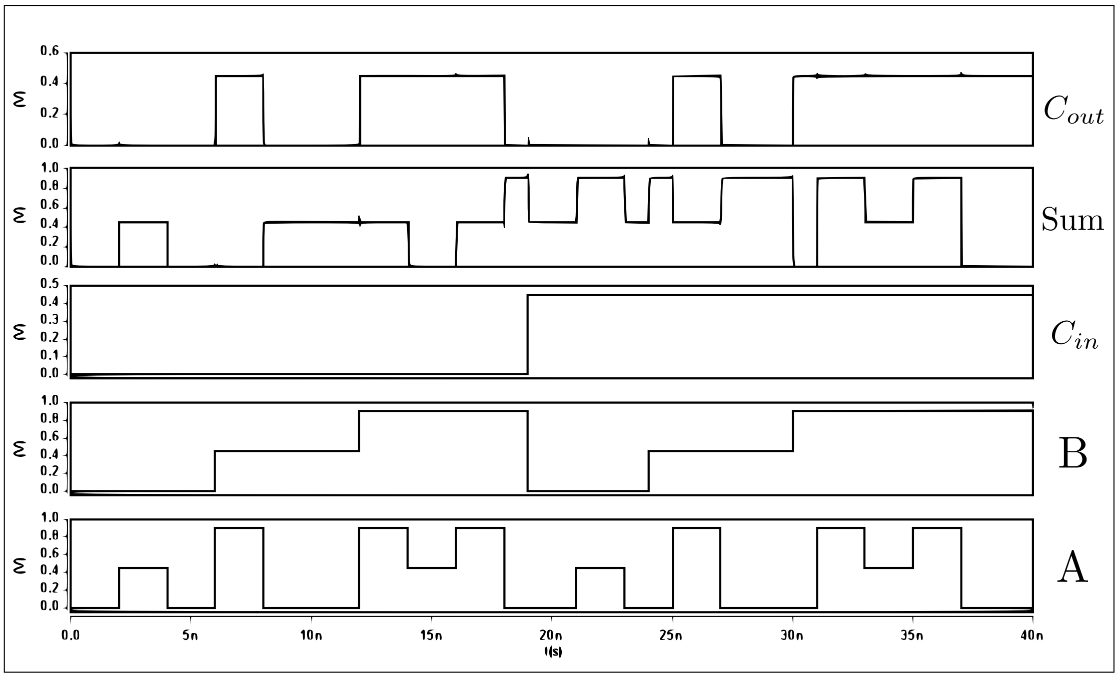


Figure 3.12: Simulation waveforms for Proposed TALU design

Table 3.5 and 3.7 shows the comparision for delay, power consumption, PDP and CNFETs count for functional modules of different TALU designs. The modules that were compared are Half Adder-subtractor, Full Adder-subtractor, 1-trit and 2-trit Multiplier. As it can be seen from the table, the proposed design PTALU shows improvement in terms of PDP when compared to other existing designs. The main reason for the PDP improvement is an improvement in power that is obtained in the proposed designs. Multiplexer based design approach for implementing ternary logic circuits is known to provide improvement in power consumption as compared to other design approaches. This is due to the fact that there are no VDD to ground paths in this approach that majorly contribute to power consumption. This multiplexer based design approach that uses 2 : 1 multiplexers was used for designing and implementing the arithmetic circuits of the proposed TALU design. The existing TALU designs, on



(a) Waveforms for Half adder



(b) Waveforms for Full adder

Figure 3.13: Waveforms for Half Adder and Full Adder of proposed TALU

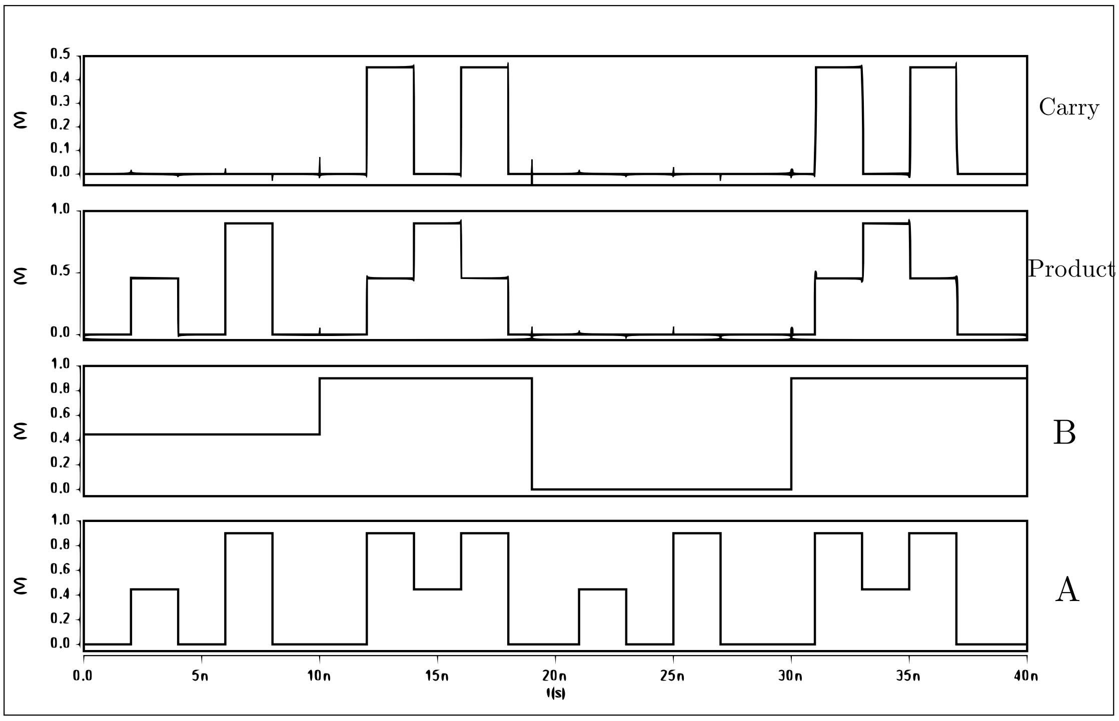


Figure 3.14: Waveforms for 1-trit Multiplier

the other hand, follow decoder-encoder based design approach which consumes more power comparatively. Percentage improvement of more than 90% is seen in PDP and power consumption in case of a Half Adder-subtractor.

The propagation delay of the proposed HAS module is less than TALU-1 but higher than TALU-2 design. Similar observation is made for FAS module and Multiplier modules, where the percentage improvement in power and PDP is more than 90%, whereas the propagation delays are comparable. For Half adder-subtractor design, other than the designs in TALU-1 and TALU-2, the adder and subtractor design in [71] is used to make a Half Adder-subtractor and Full Adder-subtractor and the results are compared to proposed PTALU. It is seen from the Table 3.5 that this design [71] gives results similar to the Adder-subtractor design in TALU-1 since both these designs are encoder-decoder based designs with some reduction in gates that is achieved in [71] by using the negation of literals technique. Also a recent work on a 1-trit multiplier design in [37] and a 2-trit multiplier design in [72] are added and compared with the proposed PTALU design. The multipliers in PTALU show considerable improvement in power upto 99% when compared to these designs due to the elimination of VDD

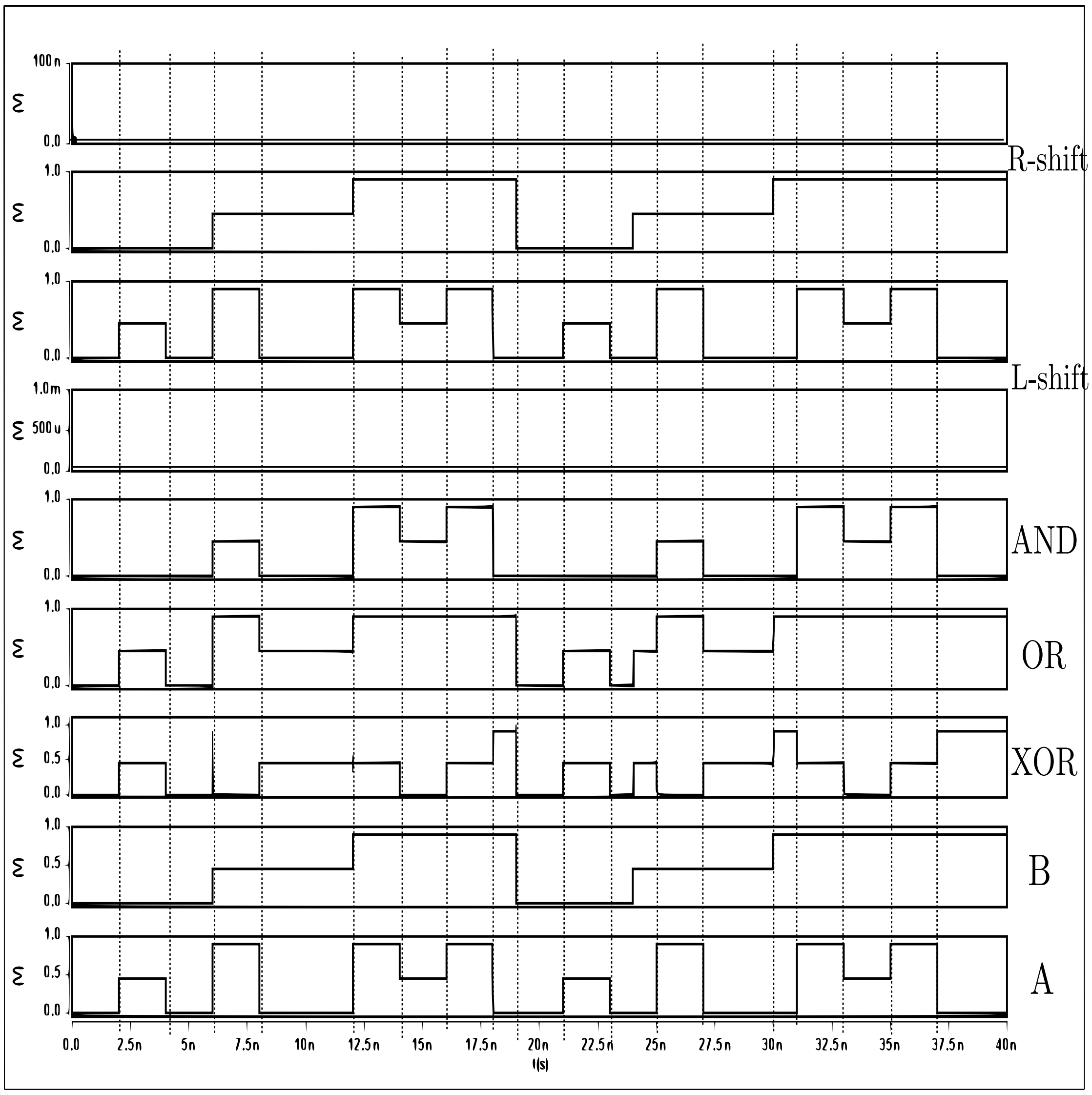


Figure 3.15: Waveforms for Logic and Shift operations

to ground paths in the proposed multiplexer based design as compared to existing multiplier designs.

An improvement is seen in the CNFETs count also in the proposed design of the functional modules. As the proposed designs are multiplexer based, they require lesser number of transistors to build the circuit as compared to encoder-decoder based designs and transistor level designs of circuits. We have not considered the CNFET count for the generation of the unary operators and PTI,NTI Muxes that are commonly used for all the modules, while calculating the count for individual functional modules. We have considered this count while calculating the CNFET count for the entire TALU design, as given in Table 3.9. In case of Full adder-subtractor module, the proposed

Table 3.7: Simulation results based Comparision of 1-trit Multiplier

Designs	Delay(ps)	Power (uW)	PDP(x10^{-18}J)	CNFETs count
TALU-1	28	0.49	13.72	56
TALU-2	9	4.05	36.45	26
[37]	5.8	56.9	330.02	26
PTALU	12.5	0.11	1.37	20
Improvement in PTALU w.r.t. TALU-1	55.3%	77.5%	90%	64.2%
Improvement in PTALU w.r.t. TALU-2	−28%	97.2%	96.2%	23%
Improvement in PTALU w.r.t. [37]	−53.6%	99.8%	99.5%	23%

design requires 44 CNFETs as compared to 342 and 112 CNFETs required for designs TALU-1 and TALU-2, respectively. Similarly for 2-digit multiplier, the proposed design is implemented using 216 CNFETs as compared to 520 and 302 CNFETs in case of the designs TALU-1 and TALU-2, respectively.

In Table 3.9, the Design-1 represents the simulation results for the two existing designs TALU-1 and TALU-2 and the proposed design PTALU without output multiplexer and the Design -2 represents the TALU-1, TALU-2 and PTALU designs with output multiplexer, respectively. Here, both the existing designs and the proposed design are given the same input test patterns and then the obtained simulation results are compared. As seen from the table, the proposed designs show up to 96% improvement in PDP, up to 91% improvement in power consumption and up to 63 % improvement in CNFETs count compared to existing designs, TALU-1 and TALU-2. The propagation delay of proposed design is less than the TALU-1 design but is comparable to the TALU-2 design. Similar observation was seen for Design-2. In case of Design-2, for building the entire TALU including the output multiplexer, the TALU-1 and TALU-2 designs require 1418 and 780 CNFETs respectively, whereas the PTALU design requires 524 CNFETs. Hence, the proposed TALU design is optimized in terms of CNFET count. This optimization in CNFET count is acheived at the TG block stage of the PTALU, by elimination of decoders, some TGs and at the functional modules

Table 3.8: Simulation results based Comparision of 2-trit Multiplier

Designs	**Delay(ps)**	**Power (uW)**	**PDP(x10^{-18}J)**	**CNFETs count**
TALU-1	880.5	19.9	17512	520
TALU-2	42.1	60.3	2538.6	302
[72]	37.6	405.2	15235.5	243
PTALU	53.1	14.9	791.1	216
Improvement in PTALU w.r.t. TALU-1	93.9%	25.1	95.4%	58.4%
Improvement in PTALU w.r.t. TALU-2	−20.7%	75.5%	68.8%	28.4%
Improvement in PTALU w.r.t. [72]	−29.1	96.3%	94.8%	11.1%

Table 3.9: Comparisions of TALU designs

TALU Designs		**Delay (ps)**	**Power (uW)**	**PDP(x10^{-17}J)**	**CNFETs count**
Design-1	**TALU-1**	83	250.8	2081.6	1370
	TALU-2	23	237.3	545.7	732
	PTALU	29	22.14	64.2	476
	Improvement in PTALU w.r.t. TALU-1	65%	91.1%	96.9%	65.2%
	Improvement in PTALU w.r.t. TALU-2	−26%	90.6%	88.2%	34.9%
Design-2	**TALU-1**	63.5	253.5	1609.7	1418
	TALU-2	28	239.3	670.0	780
	PTALU	39	25.33	98.7	524
	Improvement in PTALU w.r.t. TALU-1	38.5%	90%	93.8%	63.0%
	Improvement in PTALU w.r.t. TALU-2	−39.2%	89.4%	85.2%	32.8%

stage of the PTALU, due to use of multiplexer based design.

CNFET based TALU designs might have disadvantages such that of performance variability due to the use of various CNFETs used for design. But since we are using limited number of variations of CNFET for designing the proposed circuits, the overall effect of variations is less here.

3.5 Conclusions

This chapter presents a new design for a TernaryALU. The proposed design has function select block, transmission gate block, and multiplexer based function modules as its main components. This design eliminates the need for input decoders in the TALU architecture, using a 2 : 1 multiplexer based approach for implementing functional modules. The complexity in the TG block is also reduced in the proposed TALU design. The design is extended with an additional output multiplexer that multiplexes all the outputs obtained from the different functional modules into one single 2-trit TALU output. This multiplexing of outputs of various operations into one output can be used for TALU design in a processor. HSPICE simulations of the proposed TALU design show significant improvement in power, PDP, and the number of transistors used for the design with and without output multiplexer, with respect to existing TALU designs. This proposed 2-trit TALU design can be extended to a higher digit/trit (3-trit) TALU design by making the required changes.

Chapter 4

Design of CNFET-Based Ternary Sequential Logic circuits

4.1 Introduction

The implementation of an efficient TALU, which is a vital element of a processor is presented in Chapter 3. The sequential logic circuits are also required along with the TALU, for designing a processor. The ternary sequential logic circuits like the D-flipflops can be used to implement registers and these registers can be used to build the ternary register file used in a processor. Registers are compact, high-speed storage elements that are used to temporarily store data during program execution. They are typically constructed of numerous D flip-flops and can store a limited amount of data. Registers in a processor are used to store intermediate results, addresses, and operands, and they allow the processor to conduct computations and operations on data. D flip-flops are used to store a single bit of data in the design of registers. D flip-flops are particularly helpful since they may be used to store a logic circuit's output and hold it constant until the next clock cycle. This is vital in a processor because it synchronizes data with the processor clock, which is required for proper operation. Ternary Registers and D flip-flops create the backbone of a ternary processor's memory architecture. They allow the processor to store, retrieve, and perform operations on data.

Very few papers [57, 58, 73], propose the design of ternary sequential logic circuits such as flip-flops, latches, and counters. Mainly two design approaches are followed in existing ternary D-flipflops, one method uses STIs [58], and the other uses successor-predecessor circuits [57]. This chapter lists the various existing designs of D-flipflops using both approaches and proposes new D-flipflop designs. The fiest proposed D-flipflop design is implemented using a new Ternary buffer and STI-based approach. For the ternary D-flipflop using the successor-predecessor-based approach, an efficient design is proposed and compared to our multiplexer-based design presented in [74]. Also, two new hybrid designs for ternary D-flipflops are presented that are implemented using a combination of STIs/buffer and successor-predecessor circuits. Ternary 3-trit synchronous and asynchronous counters are also designed using all the proposed designs of the D-flip-flops. The proposed designs show significant improvement in design parameters such as power, delay and PDP compared to their existing counterparts.

4.2 Previous Work on CNFET-based Ternary Sequential Logic circuits

The ternary STI gate circuits are fundamental building blocks for the ternary sequential logic circuits such as D-latch, D-flipflop, and counters. Some designs also use ternary successor-predecessor circuits to build D-flip-flops instead of STI gates [57]. The ternary D-flip-flop and ternary counters designs existing in literature are discussed below.

4.2.1 Existing Ternary D-flipflop designs

A ternary D-latch and D-flipflop can be designed using two approaches, by building it using STI gate circuits or by using successor-predecessor circuits. Typical STI-based D-flipflop has been presented in [58]. And the successor-predecessor circuits based D-flipflops are proposed in [57] and [73]. The ternary D-flipflop is designed by connecting two D-latches in a master-slave configuration. These D-latches are in turn made up of either STI gates or successor-predecessor circuits, as shown in Figures 4.1 and 4.2, respectively. Figure 4.1 shows the designs of ternary D-flip-flops that can be implemented using STI gates. This design requires four STI gates and four transmission gates to build a D-flipflop. Each of these STI gates can be implemented using designs proposed in [5], [58] and [75], respectively as shown in Figure 4.1. The ternary D-flipflop design using a pass transistor STI gate (PT STI) is presented in [58]. The other two D-flipflop designs can be designed using the STI gates proposed in [5], which is a standard STI gate, and in [75], which is a pass transistor STI with body effect (PTB STI). The standard STI design [5] consumes large amount of power since it uses voltage division along the didode connected transistors for producing a logic '1' at the output. The pass transistor STI in [58] and [75] performs better in terms of power consumption when compared to the standard STI design since it transfers Vdd/2 to the output using pass transistors without voltage dividing.

Figure 4.2 shows the architecture of a ternary D-flipflop designed using successor-predecessor circuits. This design requires two successor circuits, two predecessor circuits, and four transmission gates to build a D-flipflop. The existing designs of ternary D-flipflops using successor-predecessor-based circuits are proposed in [57] and [73]. The paper [73] uses the names DS (Dualshift-Singleshift operators) that have the same behaviour as that of the successor-predecessor circuits. The designs for the ternary successor and predecessor circuits and successor-predecessor-based ternary D-latch, D-flipflop, and ternary counters are presented in [57]. The successor-predecessor circuits in [57], though better in terms of power than their counterparts existing in literature, have significant power consumption due to the Vdd to Gnd path created for producing a logic'1' at the output. In the Figure 4.2, under the succesor-predecessor circuits only successor circuits are shown for existing designs implemented in [57], [73] and [74] respectively. The predecessor designs are similar to the successor designs. In the successor-predecessor designs in [74] the circuits are built using 2:1 multiplexers hence called as Mux-based SP(Succesor-Predecessor). This multiplexer-based approach helps in improvement in power as compared to the existing designs in [57] and [73], as here

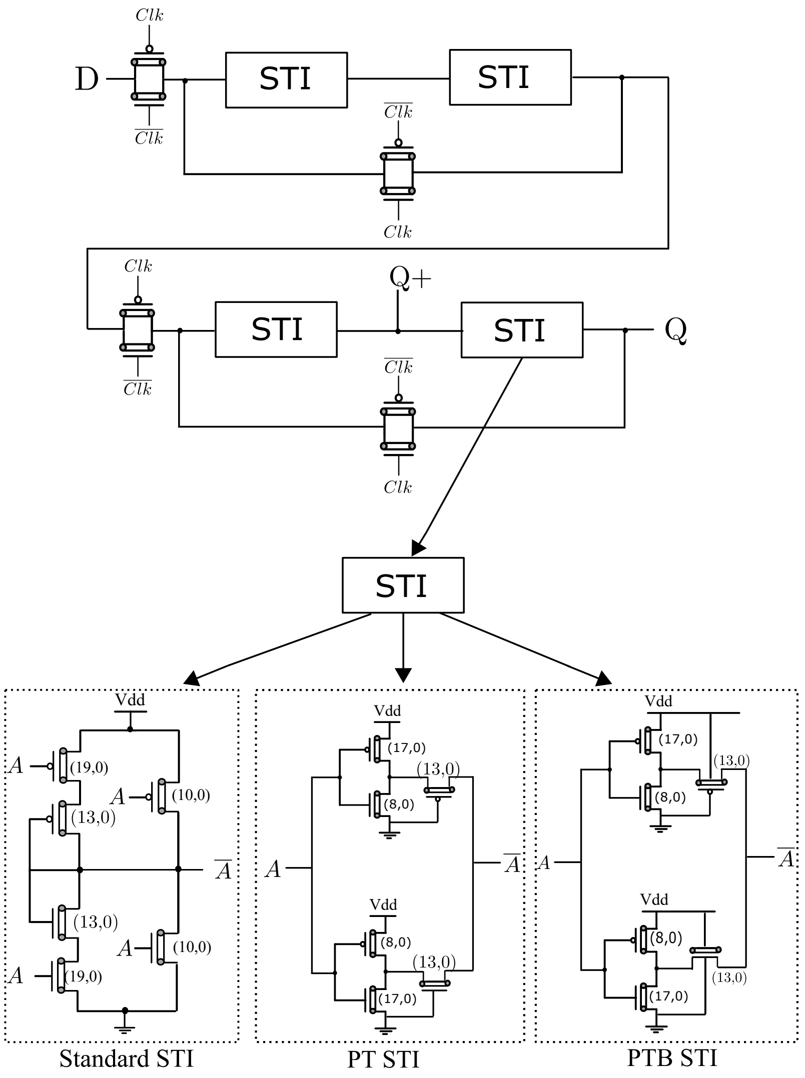


Figure 4.1: Designs of Ternary D-flipflops using existing STI gate designs of a) a standard STI [5], b) a PT-STI [58] and c) a PT-STI with body effect [75]

the Vdd to Gnd path is eliminated by passing a constant value of logic ’1’ to one of the transmission gates of the multiplexers.

All the above mentioned designs for ternary D-flipflop implementation shown in Figures 4.1 and 4.2 consume large power. To overcome this limitation, we have proposed a new architecture for implementing ternary D-flipflops using a two-supply-based ternary buffer and ternary STI. Such a design shows improvement in terms of power and PDP.

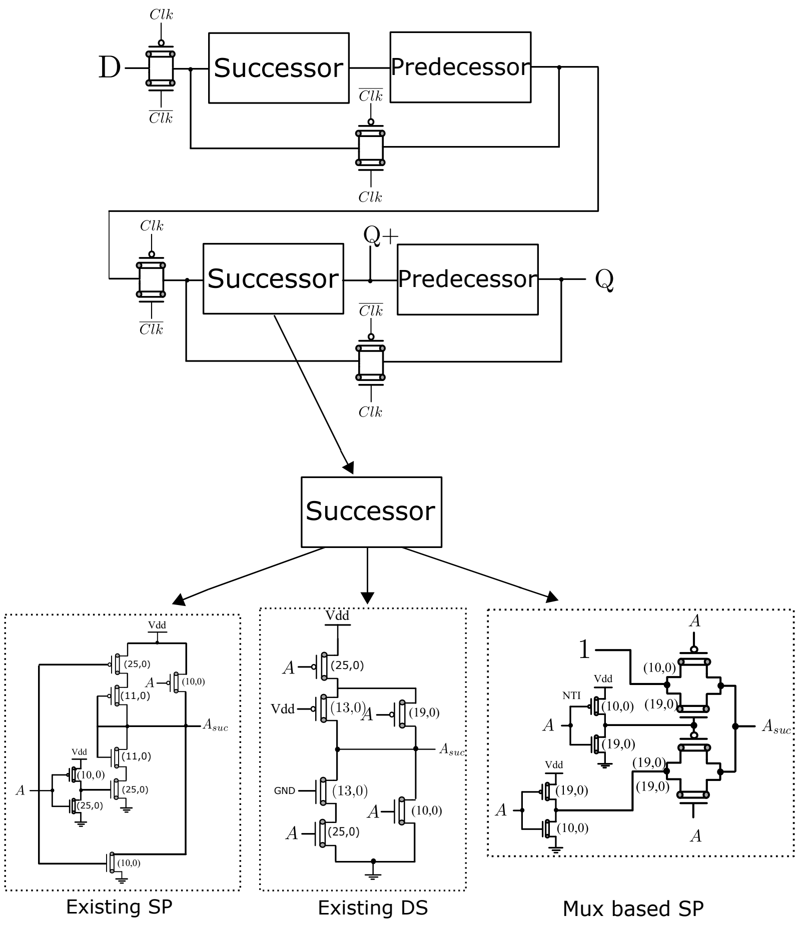
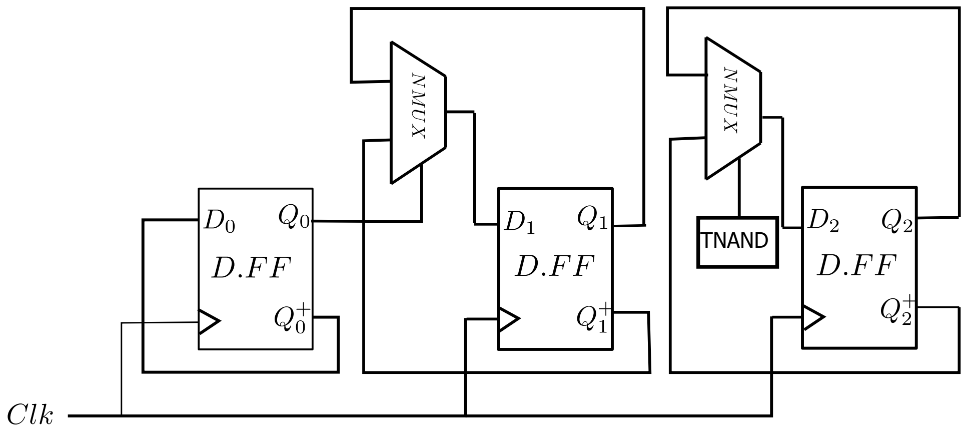
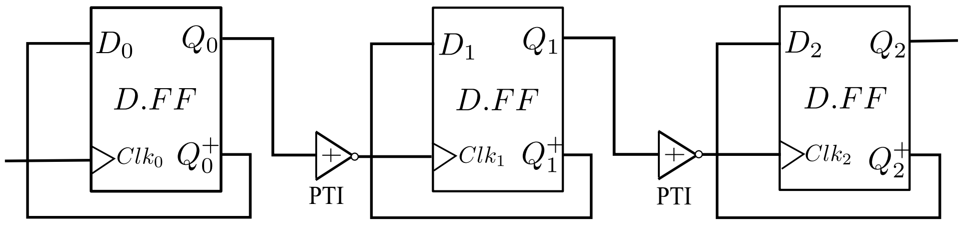


Figure 4.2: Designs of Ternary D-flipflops using Successor-predecessor gate designs in a) [57], b) [73] and c) [74]



(a) Design of 3-trit Ternary Synchronous Counter


D_0 Q_0 D.FF Clk_0 Q_0^+
D_1 Q_1 D.FF Clk_1 Q_1^+
D_2 Q_2 D.FF Clk_2 Q_2^+
PTI
PTI

(b) Design of 3-trit Ternary Asynchronous Counter

Figure 4.3: Existing design of 3-trit ternary counters in [57] and [73].

4.2.2 Existing Ternary counter designs

The above ternary D-flipflops can be used for building ternary counters. Ternary 3-trit synchronous and asynchronous counters are designed using the ternary D-flipflops in [57] and [73]. In both of these papers, the ternary counters are designed using ternary D-flipflops that are successor-predecessor based, as shown in Figure 4.2. The architecture of the ternary 3-trit synchronous and asynchronous counters is as shown in Figure 4.3. The successor-predecessor-based ternary D-flipflops present in the counters in Figure 4.3 has a Q+ output as the input's next logic state, which makes it feasible to connect the Q+ output back to the D input of the D-flip-flop and be used for ternary counters implementation. The existing counter designs proposed in [57] and [73] consume considerable power and require many CNFETs since they are constructed using the successor -predecessor circuits shown in Figure 4.2. In the next section, better designs for ternary counters are proposed using the same existing architecture but with new proposed designs of ternary D-flipflops.

4.3 Proposed designs for Ternary sequential logic circuits

This section proposes new designs for ternary sequential logic circuits like ternary D-flipflop and ternary synchronous and asynchronous counters. A new ternary buffer-STI-based architecture is proposed for a ternary D-flipflop. This design uses a power-efficient ternary buffer and STI circuits for D-flipflop implementation. A successor-predecessor-based ternary D-flipflop is also proposed that uses improved multiplexer-based successor-predecessor circuits. In addition to these designs, two new hybrid designs for ternary D-flipflops are presented in this section. Ternary 3-trit synchronous and asynchronous counters built using the proposed ternary D-flipflop designs are also presented.

4.3.1 Proposed designs for Ternary D-flipflop circuits

4.3.1.1 Proposed designs for STI-based Ternary D-flipflop circuits

The proposed design for a ternary D-flipflop is as shown in Figure 4.1. This design consists of one ternary buffer and two STI gates. The existing STI-based ternary D-flipflop consists of four STI gates as shown in Figure 4.1, two STIs in the first stage and two STIs in the second stage.A ternary buffer can replace the 2 STIs in the first stage. The two STIs in the second stage are kept as it is since we need to tap the Q+ output here for building counters. Hence, a new architecture, ternary buffer-STI based, is proposed in this section to design a ternary D-flipflop, as shown in Figure 4.4. The ternary buffer and STIs used in this D-flipflop are also presented in this work. Their circuit diagram is as shown in Figure 4.4. The proposed ternary buffer and STI are designed using dual power supplies, Vdd and Vdd/2.

The equations shown below, for an input of A, explains the behaviour of the proposed STI gate where the output $\overline{A}$ is represented in terms of the literals A^0, A^1 and A^2 as:

$$\overline{A} = 2.A^0 + 1.A^1 + 0.A^2 \tag{4.1}$$

$$\overline{A} = 2.A^0 + 1.(A^1 + A^2).(A^0 + A^1) + 0.A^2 \tag{4.2}$$

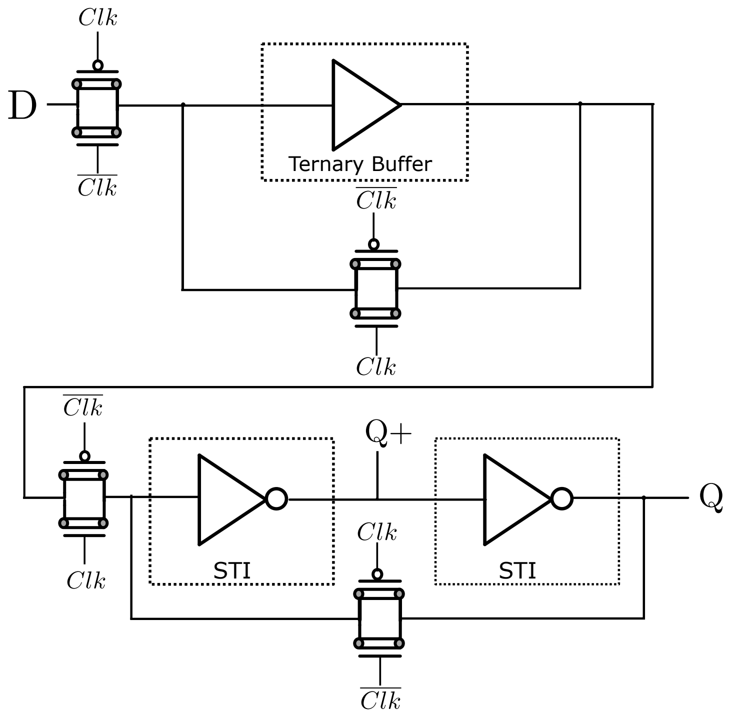
$$\overline{A} = 2.A_n + 1.\overline{A_n}.A_p + 0.A_p \tag{4.3}$$

where A^0, A^1 and A^2 are decoder outputs for A, and A_p, A_n are PTI and NTI outputs of A, as shown in Table 4.1. A^0, A^1, A^2, A_p, A_n are also called as unary operators.

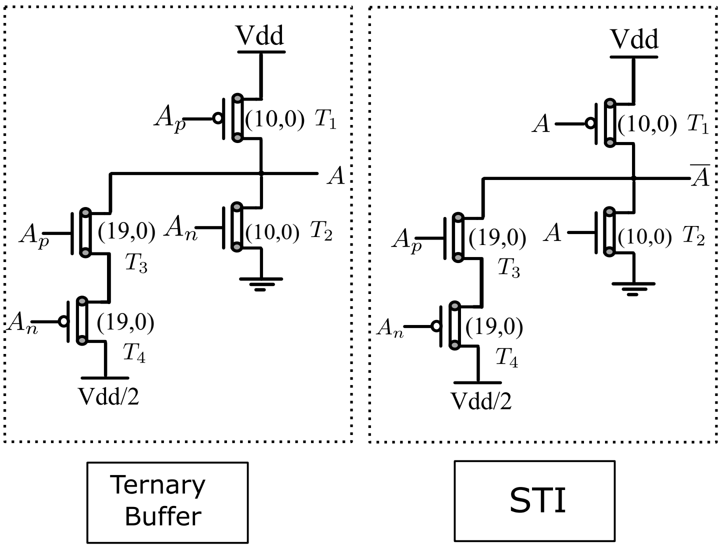
The working of the proposed STI gate with reference to the Figure 4.4

Table 4.1: Truth table for Unary operators

A	A^0	A^1	A^2	A_p	A_n
0	2	0	0	2	2
1	0	2	0	2	0
2	0	0	2	0	0



(a) Proposed Architecture of D-flipflop



(b) Proposed ternary buffer and STI circuits

Figure 4.4: Proposed Design of ternary D-flipfliop

is further explained as follows :

1) For an input of logic '0', the CNFETs T2 and T4 are OFF, the CNFETs T1 and

T3 are ON, so the output is connected to Vdd and is logic '2'. Also from Table 4.1 for input '0', only A^0 is 2, hence output is '2' from Equation 4.1 and 4.3.

2) For an input of logic '1', the CNFETs T1 and T2 are OFF, the CNFETs T3 and T4 are ON, so the output is connected to Vdd/2 and is logic '1'. Also from Table 4.1 for input 1, only A^1 is 2, hence output is '1' from Equation 4.2 and 4.3.

3) For an input of logic '2', the CNFETs T1 and T3 are OFF, the CNFETs T2 and T4 are ON, so the output is connected to Gnd and is logic '0'. Also from Table 4.1 for input 2, only A^2 is 2, hence output is '0' from Equation 4.1 and 4.3.

Similarly a ternary buffer is also designed. The circuit diagram for this is as shown in Figure 4.4. The proposed ternary buffer circuit connects to the Vdd/2 supply directly for an input of logic '1', to produce an output of logic '1'.

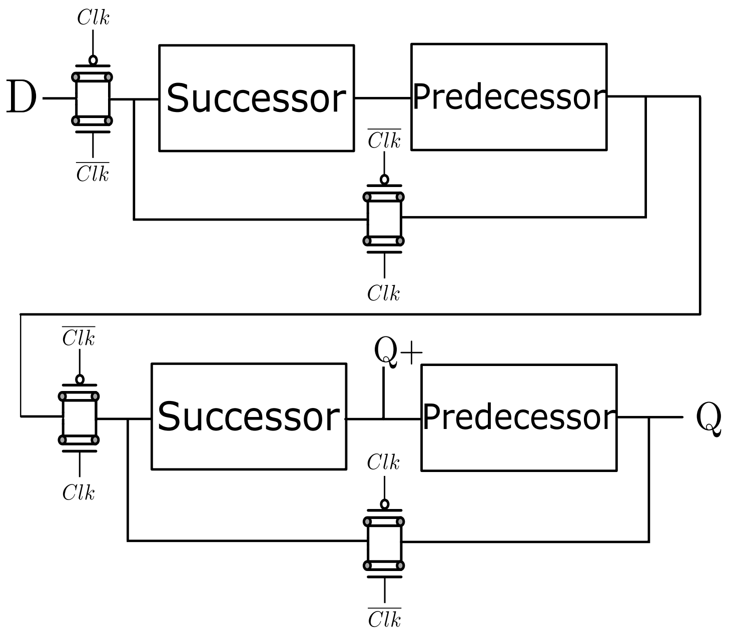
4.3.1.2 Proposed designs for Successor-predecessor-based Ternary D-flipflop circuits

A successor-predecessor based ternary D-flipflop design is proposed as shown in Figure 4.5.This is an improved version of the multiplexer-based design in [74]. The proposed successor-predecessor designs are obtained by eliminating one transistor each in the transmission gates used to pass a logic '1' in the multiplexer-based successor-predecessor circuits designs in [74]. These proposed designs for successor-predecessor circuits are used to implement the ternary D-flipflop. This design performs better than the existing successor-predecessor D-flipflop designs in terms of power and number of CNFETs. The successor-predecessor circuits used here are similar to the designs in [16].

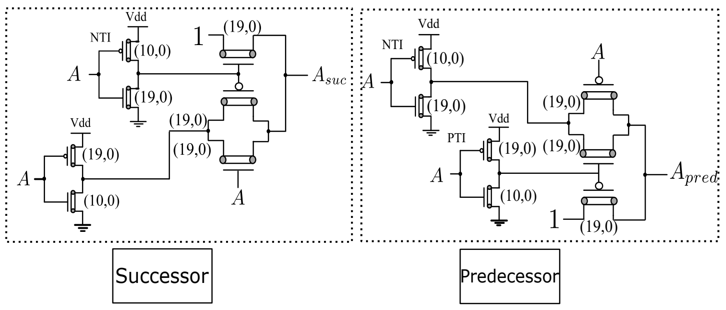
4.3.1.3 Proposed designs for Hybrid Ternary D-flipflop circuits

The existing approaches of implementing ternary D-flipflop circuits are STI based approach and successor-predecessor based approach. A third approach proposed above is a ternary buffer-STI based approach. We observe from these approaches that ternary D-flipflops can also be built by combining the above three design approaches. Using this concept, two new hybrid designs for ternary D-flipflops are proposed in this paper.

These designs are shown in Figure 4.6. The first proposed hybrid design uses a ternary buffer in the first stage and the successor-predecessor circuits in the second stage. It combines the proposed buffer-STI approach and the existing successor-predecessor approach. Here we have used the proposed ternary buffer and the best performing (among other SP-based designs) improved mux-based successor-predecessor circuits. The second proposed hybrid design uses the successor-predecessor circuits in the first stage and the STI gates in the second stage. It combines the successor-predecessor approach and the proposed buffer-STI existing approach. Here also we have used the best performing improved mux based successor-predecessor circuits and the proposed STI gate circuits. These two proposed hybrid designs for ternary D-flipflops show a performance improvement compared to the existing ternary D-flipflop designs.

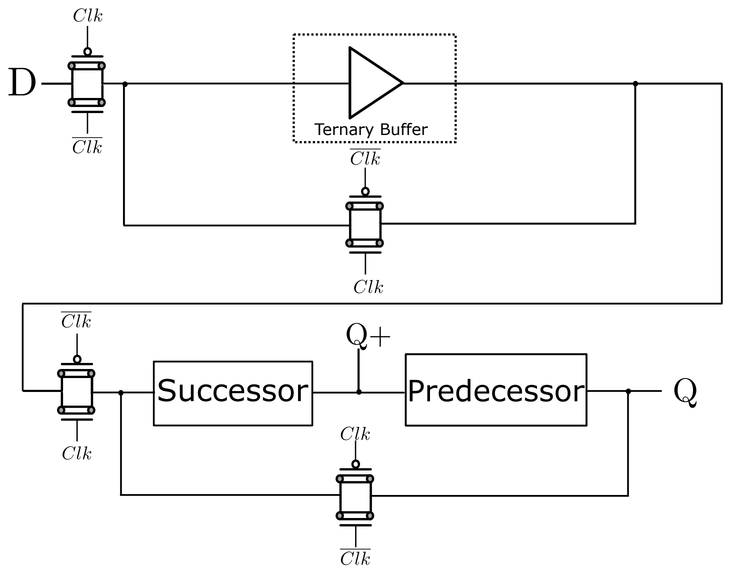


(a) Architecture of Successor-predecessor based D-flipflop

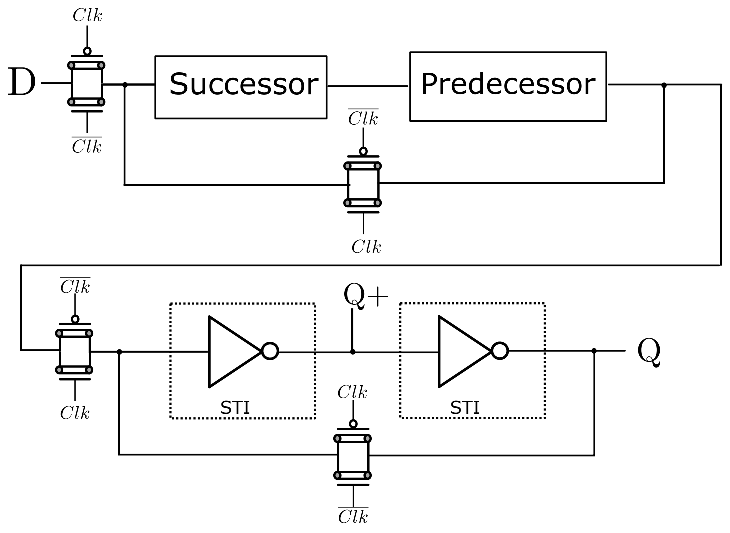


(b) Proposed design of Successor-predecessor circuits

Figure 4.5: Proposed Design of ternary D-flipfliop using successor-predecessor circuits



(a) Ternary buffer and Successor-predecessor based ternary D-flipflop



(b) Successor-predecessor and STI based ternary D-flipflop

Figure 4.6: Proposed hybrid designs for ternary D-flipflops

4.3.2 Proposed designs of Ternary counter circuits using the proposed D-flipflop circuits

The proposed designs for the ternary 3-trit synchronous and asynchronous circuits have the same architecture as that of the existing ternary counter designs as shown in Figure 4.3. But the ternary D-flipflops used in these proposed counters are the

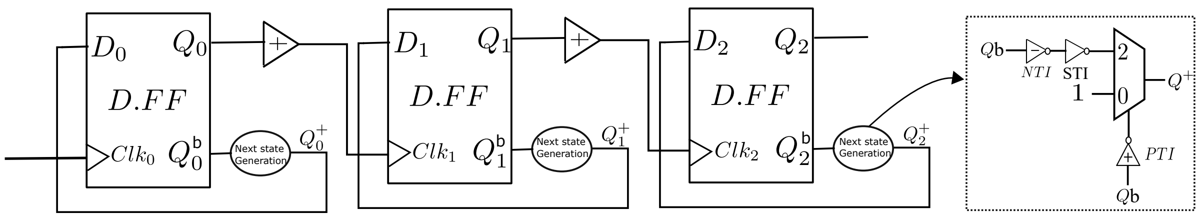


Figure 4.7: Proposed design of ternary asynchronous counters showing the next state generation logic

proposed buffer-STI based D-flipflop, improved mux-based D-flipflop and hybrid D-flipflops. So, we obtain four proposed designs for counters each using one proposed D-flipflop design.

As we can see from the architecture of the existing counters in Figure 4.3, for each D-flipflop the output Q+ which is the next state,successor output (since it is a SP based D-flipflop) is fedback to the input so that proper counting happens. This behaviour is not possible to implement if we use the proposed buffer-STI based flipflop in the proposed counter designs since the Q+ output of this flipflop is STI inversion of the input and not the next state of the input as desired. Similarly for the first hybrid D-flipflop design it is possible to implement a counter since the second stage is successor-predecessor circuit giving us the Q+ output as next state but for the second hybrid design since the second stage is STI gates the Q+ output is the STI inversion. Hence when used directly for building counters, the proposed buffer-STI based D-flipflop and the second hybrid D-flipflops do not work. So for these counters, there is a need of extra circuitry at the Q+ output of each of the three D-flipflops used for building both 3-trit ternary synchrnous and asynchronous counters. This extra circuit called as the next state generation logic is as shown in the Figure 4.7 . This circuit takes Qb (STI inverted value) as input and gives a Q+ (next state value/successor output) as output as shown in the Truth table 4.2. It can be implemented using a 2 : 1 multiplexer, a NTI, a PTI and an inverter according to the Equation 4.4.

$$\overline{Q+} = 1.\overline{Qb_p} + Qb.\overline{Qb_n} \tag{4.4}$$

Table 4.2: Truth Table for next state generation logic

Qb	**Q+**
0	0
1	2
2	1

The output of the 2 : 1 multiplexer is then fedback to the input D of the flipflop as desired to obtain correct counter finctionality. The Figure 4.7 shows the proposed 3-trit ternary asynchronous counter that uses the next state generation logic circuit in the feedback path of the proposed STI based D-flipflop. Similarly, the 3-trit ternary synchronous counter is also designed using proposed STI based D-flipflop using the next state generation logic circuit that is connected at the output of each flipflop.

In all, three designs each for 3-trit ternary synchronous and asynchronous counters are proposed. One using the first hybrid D-flipflops, the other using the buffer-STI based D-flipflops with extra circuitry and the third using the second hybrid D-flipflops with extra circuitry.

In this section, simulation results for the proposed and existing designs of CNFET-based ternary sequential logic circuits are presented. Ternary D-flipflops and counters are realized and circuit parameters are compared for relative performance. The simulation environment used and the results obtained are discussed in the following sections.

4.4 Simulation Results

In this section, simulation results for the proposed and existing designs of CNFET-based ternary sequential logic circuits are presented. Ternary D-flipflops and counters are realized and circuit parameters are compared for relative performance. The simulation environment used and the results obtained are discussed in the following sections.

4.4.1 Simulation Environment

Simulations are done using HSPICE, a simulation tool by Synopsys. The CNFET model of [21,22] is used at a power supply of 0.9V and room temperature. The CNFETs used in the simulations are configured to have three tubes and a default pitch value of 20 nm. Power consumption results are obtained by simulating the circuits with random input patterns. The inputs are similar for the existing and proposed circuits for a fair comparison of circuit parameters. The propagation delay and PDP of the designs are also measured and compared. The simulation results for the proposed circuits are obtained for variations in load and temperature as shown in the next section.

4.4.2 Results and Discussion

4.4.2.1 STI Gate

The STI gate behaves as shown in its truth table in 4.1. Various STI gate designs have been proposed in the literature so far. The proposed STI gate design shows better performance as compared to existing STI gates in terms of power, delay, and PDP. For all the designs the propagation delay was calculated by connecting a 1fF load at the output of the STI gate and power was calculated by giving the same random input pattern to all the designs. The proposed STI gate, as shown in the Table 4.3, shows an improvement of up to 88% in power as compared to STI-1 [76], up to 71% in delay as compared to STI-2 [58] and up to 87% in PDP as compared to STI-4 [19]. Though the propagation delay of the proposed STI is a bit larger than a few of the existing designs, the circuit performs best in terms of power consumption. This improvement in power is mainly due to the use of two power supplies Vdd and Vdd/2, where the output connects directly to the Vdd/2 supply and disconnects to other paths of the circuit for the input of logic '1' hence producing an output of logic '1' as shown in 4.3.1.

The Design-1 [76] has a direct Vdd to Gnd path for producing an output of logic '1' that accounts for power consumption. The designs STI-3 [77] and STI-4 [19] also use a Vdd/2 power supply but incase of STI-3 [77] there still exists a Vdd to Vdd/2

Table 4.3: Simulation results comparision of existing and proposed STI gates

Design	Power(uW)	Delay(ns)	PDP
STI-1 [76]	0.30	25.1	7.53
STI-2 [58]	0.15	50.9	7.63
STI-3 [77]	0.29	36.7	10.64
STI-4 [19]	0.049	138.8	6.80
STI-5 [75]	0.10	52.9	5.29
Proposed STI	0.034	39.3	1.33

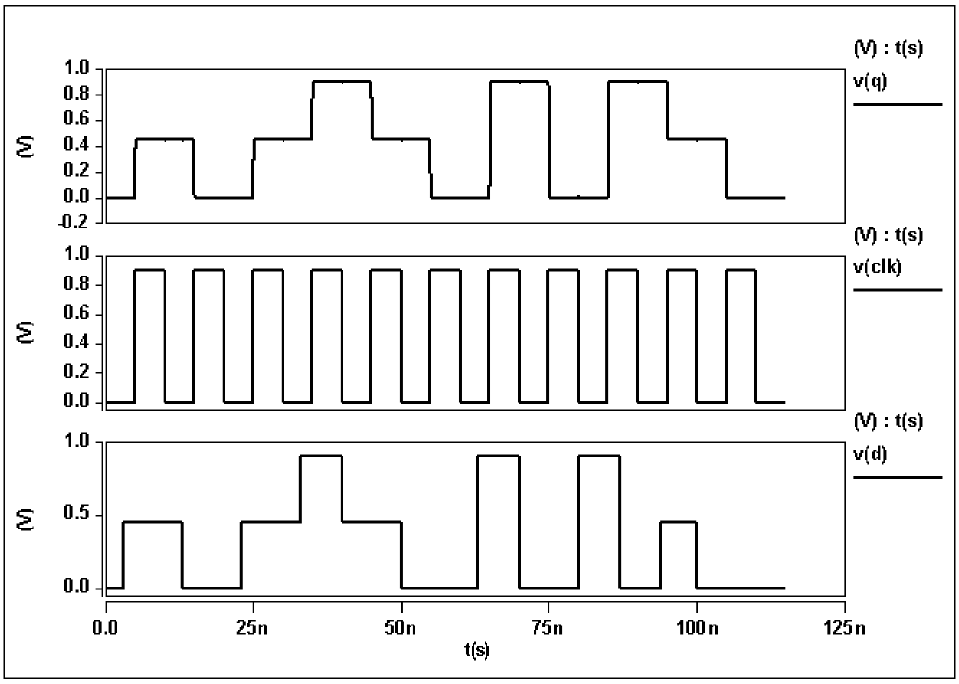


Figure 4.8: Waveforms for proposed buffer-STI based ternary D-flipflop

path that accounts for power and in the case of STI-4 [19] the delay is large as it uses a large number of transistors for the STI design. The designs STI-2 and STI-5 use pass transistor in the STI design for generating the output, avoiding the Vdd to Gnd path but still consume considerable power. The proposed STI design outperforms the rest in terms of PDP and power consumption. As the proposed STI gate shows better performance than the existing STI gates, in this paper, the proposed STI is used for building ternary D-flipflops as shown in Figure 4.4, which are found to be better in performance as compared to their existing counterparts. The proposed ternary buffer is also designed using two power supplies similar to the proposed STI.

Table 4.4: Naming convention of various existing and proposed ternary D-flipflop designs

Design	**Remark**
DFF-1	Ternary D-flipflop using the STI in [5]
DFF-2	Ternary D-flipflop in [58]
DFF-3	Ternary D-flipflop using the STI in [75]
DFF-4	Ternary D-flipflop in [57]
DFF-5	Ternary D-flipflop in [73]
DFF-6	Ternary D-flipflop in [74]
Proposed Design-1	Proposed ternary buffer and STI based D-flipflop
Proposed Design-2	Proposed successor-predecessor based D-flipflop
Proposed Design-3	Proposed hybrid design-1 for D-flipflop
Proposed Design-4	Proposed hybrid design-2 for D-flipflop

4.4.2.2 Ternary D-flipflop

The proposed ternary buffer-STI based D-flipflop and the hybrid D-flipflop designs are simulated and the performance parameters are compared to the existing designs. The existing papers that talk about ternary D-flipflop designs are the STI based flipflop in [58] and the successor-predecessor based flipflops in [57, 73, 74]. Here along with these three existing ternary D-flipflop designs we have also implemented the ternary D-flipflops by using a standard STI [5], a recent design of STI [75] since it is possible to design D-flipflops using these STI designs as well. All the designs are given same input pattern so that there is fair comparision and the performance parameters like power, delay and PDP are compared. The naming convention used for various ternary D-flipflop designs that are implemented for comparision is listed in Table 4.4.

The Table 4.5 shows the comparison of performance for various ternary D-flipflop designs. There are ten designs of ternary D-flipflops that are compared in total. Six out of these ten are existing designs and the rest four are proposed designs. The Table 4.5 shows the power in microwatts, propagation delay in picoseconds and PDP for all the ternary D-flipflop designs. The power consumed in microwatts and the respective percentages when compared to the DFF-1 design are presented in the first column of the Table 4.4. Among all the designs, we have assumed the power consumed by the DFF-1 design to be a 100% since this consumes the maximum power. It is observed that the

Table 4.5: Simulation results comparision of various existing and proposed ternary D-flipflop designs

Design	**Power(uW)**	**Delay(ps)**	**PDP**	**CNFETs count**
DFF-1	1.13(100%)	78.7	88.931	34
DFF-2	0.49(43.4%)	132.6	64.974	34
DFF-3	0.26(23.0%)	137.5	35.75	34
DFF-4	0.25(22.1%)	74.8	18.7	42
DFF-5	1.02(90.2%)	83.6	85.272	34
DFF-6	0.146(12.9%)	94.1	13.7386	42
Proposed Design-1	0.066(5.8%)	73.2	4.831	34
Proposed Design-2	0.106(9.3%)	84.3	8.9358	38
Proposed Design-3	0.091(8.05%)	81.5	7.4165	32
Proposed Design-4	0.083(7.3%)	72.3	6.0009	40

Proposed Design-1 consumes the least amount of power compared to all other designs, that is only 5.8. Similarly the other proposed designs, the successor-predecessor based Proposed Design-2 and the proposed hybrid designs, Proposed Design-3 and Proposed Design-4 consume only 9.3% , 8.05 and 7.3 of power compared to DFF-1 design. The main reason behind the power savings is the use of two power supplies in all of the proposed designs as compared to most of the existing designs that consume more power due to the presence of Vdd to Gnd path required for producing an output of logic '1'.

The total number of CNFETs required to build the ternary D-flipflops are also shown in the last column of Table 4.5. It is observed that the Proposed Design-3 which is a hybrid design required only 32 transistors that is the least transistor count among all the D-flipflop designs.

The Proposed Design-1 which is built using a two supply based ternary buffer and two STI gates not only consumes least amount of power but also the propagation delay of this design is the lowest when compared to all other designs. Hence the PDP of this design is also the lowest at 4.83. The PDP of all the four proposed ternary D-flipflop designs is much less than the existing D-flipflop designs. The Proposed Design-1 that performs the best among the four proposed designs, shows an improvement of upto 86.5%, 73.2% and 93.5% in power when compared to the ternary D-flipflop designs in DFF-2 [58], DFF-4 [57] and DFF-5 [73] respectively. The Proposed Design-1 also

Table 4.6: Simulation results of proposed ternary D-flipflops with Vdd/2 generation path

Design	**Power(uW)**	**Delay(ps)**	**PDP**
Proposed Design-1	0.176	132.2	23.26
Proposed Design-2	0.42	145.7	61.19
Proposed Design-3	0.68	147.1	100.02
Proposed Design-4	0.38	129.2	49.02

shows an improvement in PDP of upto 92.5%, 74.1% and 94.3% when compared to designs in DFF-2 [58], DFF-4 [57] and DFF-5 [73] respectively.

The Figure 4.8 shows the waveform for the buffer-STI based proposed ternary D-flipflop. Similar waveforms are observed for the other proposed ternary D-flipflop designs as well.

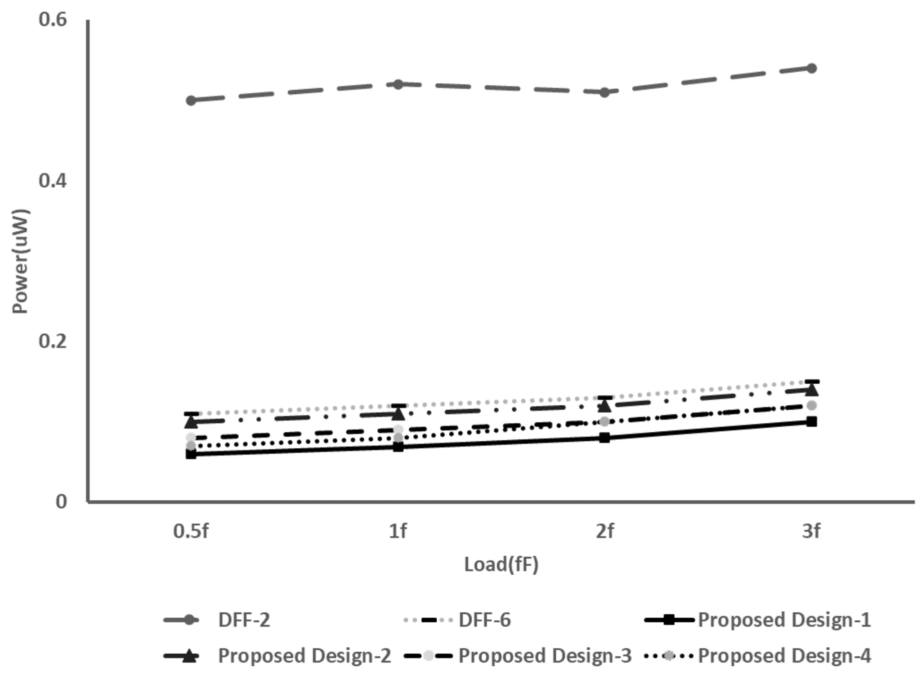
Since all the four proposed ternary D-flipflop designs use two power supplies Vdd and Vdd/2, it is assumed that two power supplies are present and hence are used. Considering that only one supply is present that is Vdd and we need to produce a Vdd/2 supply from the Vdd supply itself, we have also implemented all the four ternary D-flipflop designs by connecting them to a Vdd/2 generation path. This Vdd/2 generation path that gives the best power compared to other existing designs is that of the one in [75]. The simulation results for these proposed ternary D-flipflop designs using the Vdd/2 generation path in [75] are shown in Table 4.6 . It can be observed from the Table that the first Proposed design even after including the Vdd/2 path generation, shows better power results as compared to existing D-flipflop designs in DFF-2 [58], DFF-4 [57] and DFF-5 [73]. Though the delay for these designs is more compared to other existing and proposed designs in Table 4.4, the Proposed design-2 with Vdd/2 path included is better in power than DFF-2 [58] and DFF-5 [73], as shown in Table 4.6. Similarly the hybrid designs with Vdd/2 path Proposed design-3 is better in power than DFF-2 [58] and Proposed design-4 is better than DFF-2 [58] and DFF-5.

Simulations are also performed for the existing and proposed designs of ternary D-flipflops for variations in load, temperature and supply. The performance in terms of power and delay of various flipflop designs for variations in load capacitance is as shown in Figure 4.9. The load capacitance is varied from $0.5f - 1f - 2f - 3fF$. The power does not change much for variations in output load capacitance from low to high, it just increases slightly. Also, we see from the graph that the proposed designs consume low power even for varying loads compared to the existing designs. The delay for the proposed designs for varying loads is comparable to the design DFF-6 but less than the STI-based design DFF-1. The performance in terms of power and delay of various flipflop designs for variations in temperature is as shown in Figure 4.10. The temperature is varied from $27-50-75-120-175^0C$. As the temperature increases the power consumption increases and delay decreases as seen in the graph. It also shows that the power and delay for the proposed D-flipflop designs are less than the existing designs of DFF-2 and DFF-6 for different temperature values. The performance in terms of power and delay of various flipflop designs for variations in supply voltage is as shown in Figure 4.11. The supply is varied from $0.8 - 0.9 - 1V$. As supply voltage increases the circuit draws more current hence power consumption increases and propagation delay decreases. This behaviour is observed for all D-flipflop designs in the graph. Also for the proposed D-flipflop designs the power consumed is lesser at different supply voltages than the existing designs.

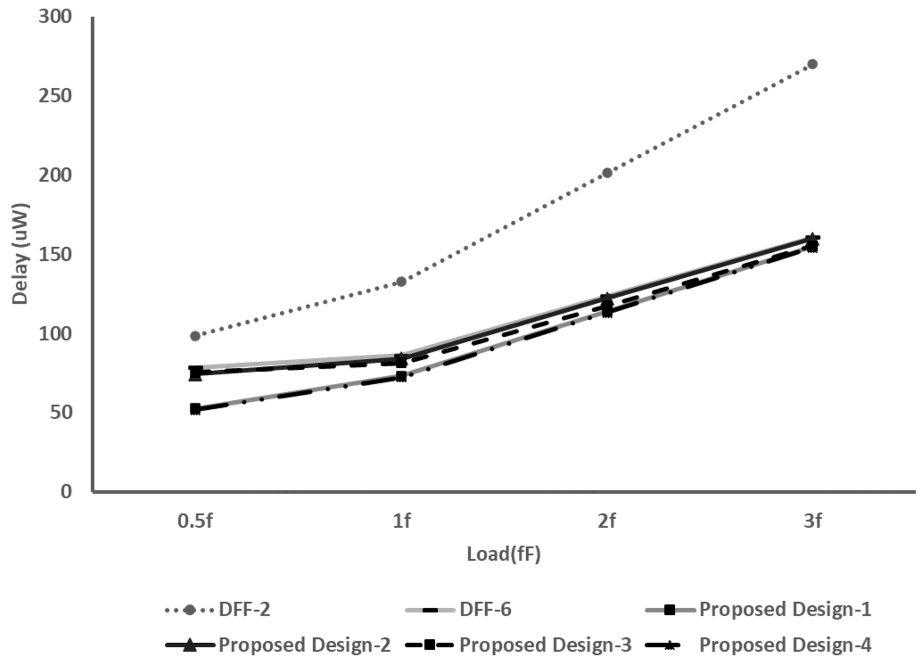
Monte Carlo simulations are also done for the existing and the proposed ternary D-Flipflop designs. The variation in propagation delay and power is calculated for variations in diameter of the CNFET. the analysis is shown in Figure . Here the existing ternary D-FF is the DFF-5 design and the proposed D-FF is the proposed successor-predecessor based DFF, Proposed Design-2. It is observed that the effect of diameter variation has less effect on proposed designs when compared to existing designs.

4.4.2.3 Ternary Counter

The ternary 3-trit synchronous and asynchronous counters are presented in [57, 73]. Using this architecture in Figure 4.3 for building counters using D-flipflops, the ternary counters are implemented and simulated using the various ternary D-flipflop designs shown in Figures 4.1 and 4.2 respectively. Similarly the ternary counters proposed in

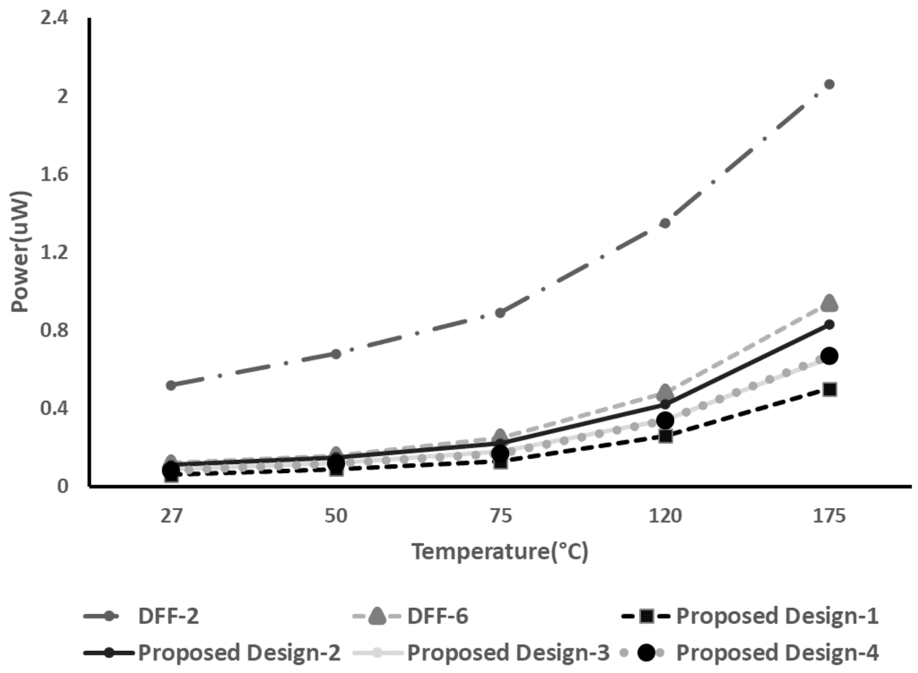


(a) Power vs Load

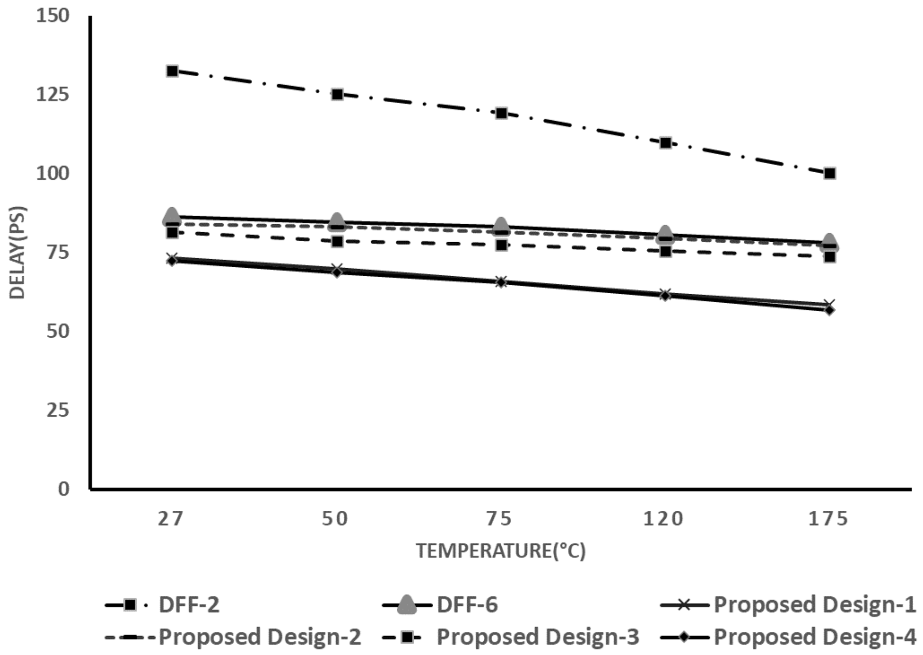


(b) Delay vs Load

Figure 4.9: Comparision of performance of various designs of ternary D-flipflop for variations in load in terms of a) power b) delay

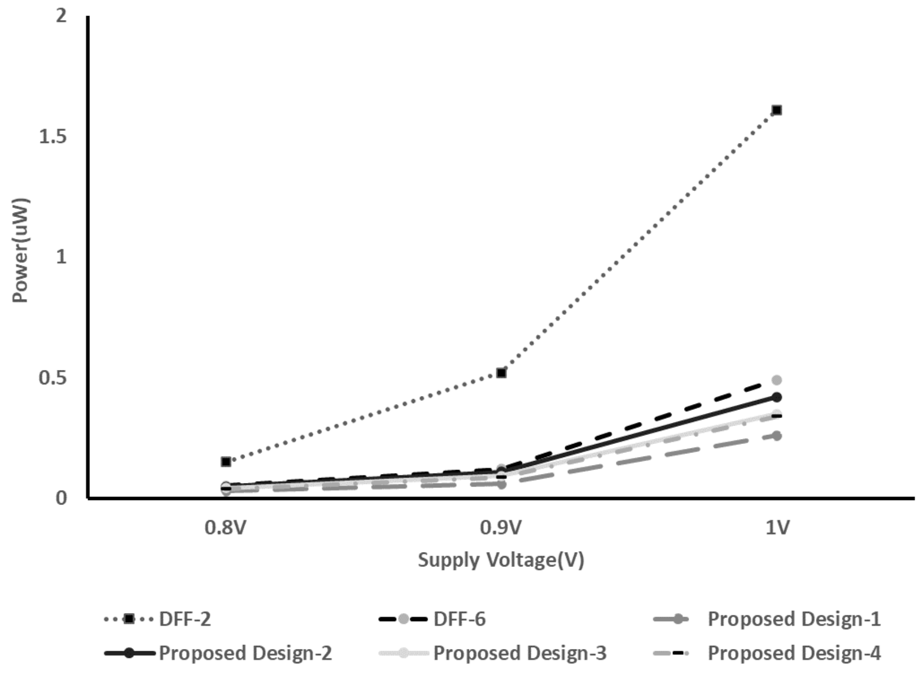


(a) Power vs Temperature

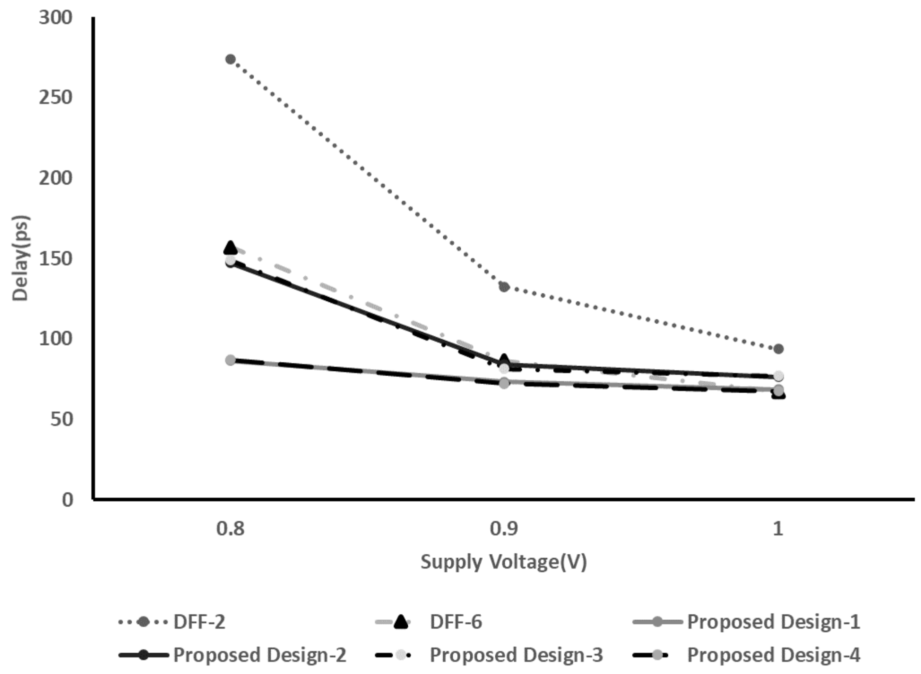


(b) Delay vs Temperature

Figure 4.10: Comparision of performance of various designs of ternary D-flipflop for variations in temperature in terms of a) power b) delay

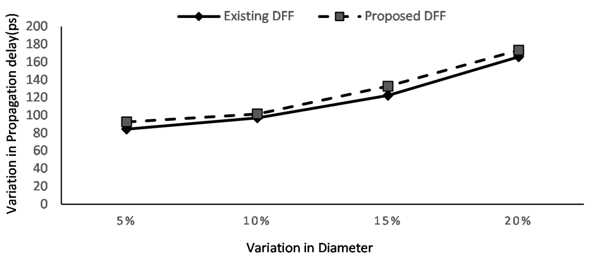


(a) Power vs Supply voltage

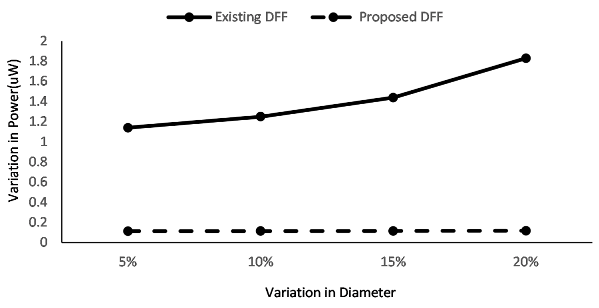


(b) Delay vs Supply voltage

Figure 4.11: Comparision of performance of various designs of ternary D-flipflop for variations in supply voltage in terms of a) power b) delay



(a) Variation in Propagation delay(ps) vs diameter



(b) Variation in Power(uW) vs diameter

Figure 4.12: Effect of variation in diameter on propagation delay and power

this paper implemented using the proposed ternary buffer-STI based D-flipflop and the hybrid flipflops are simulated using HSPICE. The simulation results are compared for all the existing six designs and proposed four designs for both 3-trit ternary synchronous counters and 3-trit asynchronous counters. 3-trit refers to the fact that this counter can count upto three ternary digits.

These results are presented in Table 4.7 and 4.8. These tables shows the power, delay and PDP for various ternary counter designs in the columns and the various ternary D-flipflops used to build these counters in the rows. It is seen from the table that for ternary synchronous counter implemented using the proposed buffer-STI based design, Proposed Design-1, consumes the least amount of power that is just 8.9% if we consider that the existing designs DFF-1 ans DFF-5 consume the maximum power which is assumed to be a 100%. Similarly, the Proposed Design-2 and hybrid designs consume just $8 - 10\%$ of the power. As the proposed designs consume low power, their PDP is also less than the existing designs. The proposed designs of D-flipflops when used for ternary asynchronous counter designs also consume just $7-8\%$ of power assuming the maximum power consumed by its existing counterpart is 100%. It can be concluded that among all the counter designs the one using buffer-STI based D-flipflop

Table 4.7: Simulation results comparision of various existing and proposed 3-trit ternary synchronous counter designs

Design	**Ternary Synchronous counter**			
	Power(uW)	**Delay(ps)**	**PDP**	**CNFETs count**
DFF-1	4.14(100%)	91.1	377.15	136
DFF-2	1.4(33.8%)	194.7	272.58	120
DFF-3	0.74(17.8%)	334.0	247.16	136
DFF-4	0.86(20.7%)	72.2	62.092	144
DFF-5	4.14(100%)	71.5	296.01	120
DFF-6	0.53(12.8%)	82.7	43.83	144
Proposed Design-1	0.37(8.9%)	68.12	25.20	136
Proposed Design-2	0.418(10%)	77.1	32.22	132
Proposed Design-3	0.37(8.9%)	77.3	28.60	114
Proposed Design-4	0.416(10%)	69.2	28.78	154

Table 4.8: Simulation results comparision of various existing and proposed 3-trit ternary asynchronous counter design

Design	**Ternary Asynchronous counter**			
	Power(uW)	**Delay(ps)**	**PDP**	**CNFETs count**
DFF-1	3.1(100%)	89.5	277.45	122
DFF-2	1.28(41.2%)	243.8	312.06	106
DFF-3	0.95(30.6%)	254.3	241.58	122
DFF-4	0.61(19.6%)	68.9	42.02	134
DFF-5	3.1 (100%)	63.8	197.7	106
DFF-6	0.42(13.5%)	75.2	31.58	134
Proposed Design-1	0.22(7.09%)	65.7	14.4	122
Proposed Design-2	0.26(8.3%)	74.9	19.4	118
Proposed Design-3	0.22(7.09%)	75.2	16.54	100
Proposed Design-4	0.25(8.06%)	66.1	16.52	140

(Proposed Design-1) performs the best in terms of power consumption and PDP. This counter shows an improvement of up to 59.03% and 91.06% in power and up to 59.41% and 91.4% in PDP when compared to ternary synchronous counters designed in [57] and [73] respectively. Similarly the proposed asynchronous counter built using the Proposed Design-1 shows an improvement of up to 63.9% and 92.9% in power and up to 65.7% and 92.7% in PDP when compared to ternary asynchronous counters designed in [57] and [73] respectively. The transistor count is also presented for the ternary counters in the last columns of Tables 4.7 and 4.8. It is observe that the counters build using the Proposed design-3 D-flipflops require 114 and 100 tranistors which is the least number of transistors for ternary 3-trit Synchronous and Asynchronous counters respectively compared to all other counter designs.

4.5 Conclusions

In this chapter, a complete review of all existing ternary D-flipflops and ternary counters is presented along with its comparison with new proposed designs. Four new proposed designs of ternary D-flipflops are implemented. The first design is a ternary buffer-STI-based D-flipflop and the second is an improved version for a multiplexer-based successor-predecessor D-flipflop. Two hybrid designs are proposed which are a combination of ternary buffer, successor-predecessor circuits and/or STI gates. The proposed designs use the ternary buffer and STI which are two supply -based designs that consume very little power. Among all the existing and proposed designs for ternary D-flipflops, the ternary buffer-STI-based D-flipflop performs the best in terms of power consumption and PDP. Ternary 3-trit synchronous and asynchronous counters are also designed using the proposed D-flipflop designs and their simulation results are compared to existing ternary counter designs. Hence, new designs for ternary sequential circuits like ternary D-flipflops and ternary counters are proposed in this work that shows considerable improvement in circuit performance parameters like power(up to 93% for D-flipflop and up to 92% for counter) and PDP (up to 94% for D-flipflop and up to 92% for counter) compared to existing designs.

Chapter 5

Design of CNFET-based Ternary Memory

5.1 Introduction

SRAM (Static Random Access Memory) and registers are two types of memory elements used in processor design. SRAM is a form of memory used to store huge amounts of data, usually as cache memory. SRAM is slower than registers, but it is more denser, which means it can store much more data in a much smaller space. In general, registers are used to store data that must be retrieved fast and frequently, whereas SRAM is used to store greater amounts of data that may not require a frequent access. The ternary SRAM design proposed in this Chapter is used in the design of ternary Instruction and Data memory in the design of the ternary processor in Chapter 6.

Memory is an essential part of any processor as it constitutes a major part of the processor and hence low power design of SRAM is desired for high performance. A ternary SRAM holds three logic states of $0, 1$ and 2. CNFET based implementation of ternary SRAMs has been found to be efficient. The storage element in a ternary SRAM conventionally constitutes of two back to back STI gates. The cycle-operators or the ternary buffer can also be used as storage element instead of the two STI gates in SRAM design.

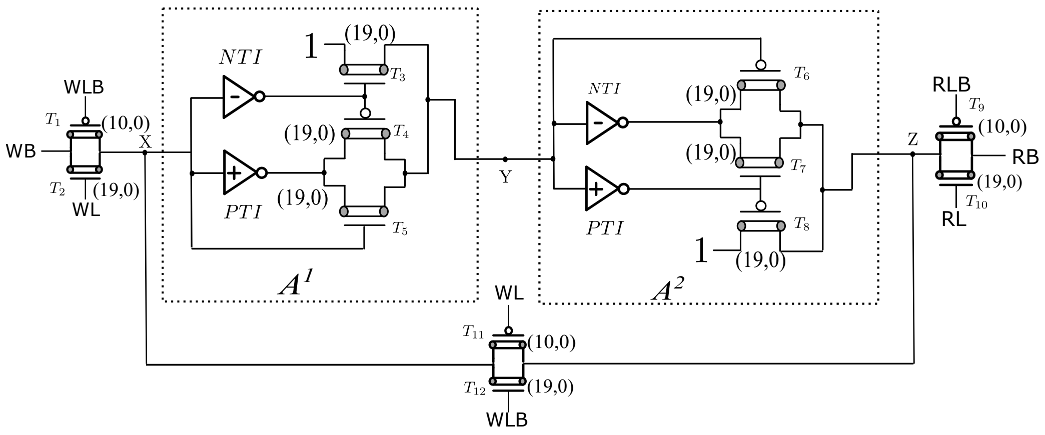
This chapter presents two new designs of cycle-operator -based ternary SRAMs and one new design of a ternary buffer-based SRAM. The architecture of the ternary SRAM cell is similar to the design in [16]. All the three designs use cycle-operators and ternary buffer that are designed using two power supplies of Vdd and Vdd/2. This dual supply-based design approach consumes low power and area. The Section 5.2 presents the existing designs and the Section 5.3 presents the proposed designs for ternary SRAM. The Section 5.4 talks about the simulation results of the proposed ternary SRAM cell designs and their comparision with the existing designs. Section 5.5 concludes the chapter.

Table 5.1: Truth Table for A^1,A^2, $A_p$and A_n

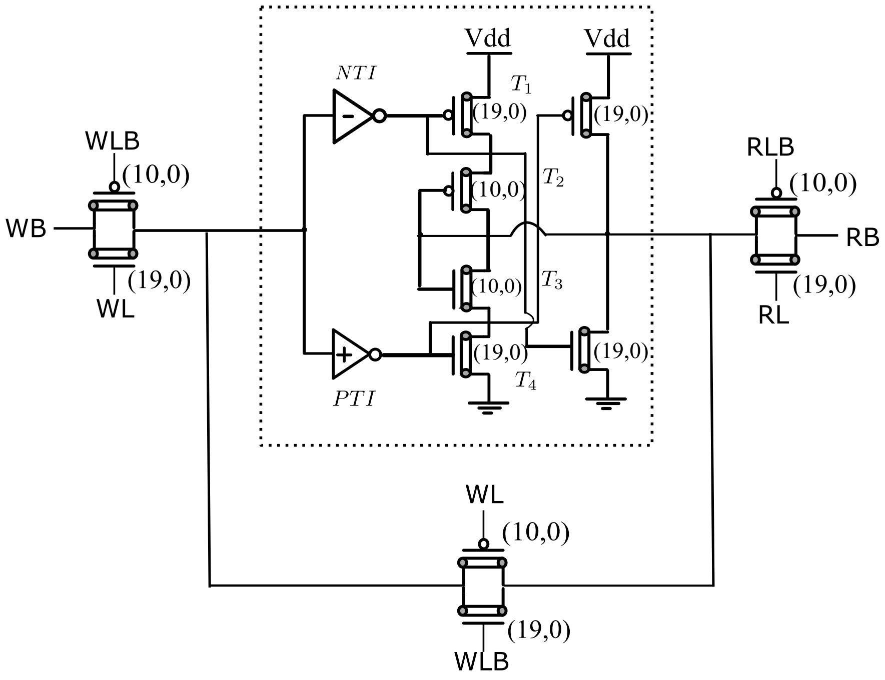
A	A^1	A^2	A_p	A_n
0	1	2	2	2
1	2	0	2	0
2	0	1	0	0

5.2 Review of Ternary Memory

The very first CNFET-based ternary SRAM design using STIs was proposed in [17]. Few other ternary SRAM designs are proposed in [15,64,65]. All these existing designs are STI-based. A latest design in [16], shows the design of ternary SRAM that uses cycle-operators (A^{1}and A^{2}as shown in Table 5.1) and a ternary buffer instead of STIs as storage elements. This design methodology for building ternary SRAM has proved to be much more efficient in terms of power and performance when compared to the standard STI-based method.



(a) Design of cycle operator-based ternary SRAM [16]



(b) Design of buffer-based ternary SRAM [16]

Figure 5.1: Existing designs of ternary SRAM [16]

The Figure 5.1 shows this existing cycle operator -based and the ternary buffer-based SRAM cell designs. The working of the cycle-operator based SRAM in Figure 5.1 (a), is as explained below:

1) The writing onto the SRAM starts when WL is high.

2) The data is passed from WB line from the transmission gate pair $T_1 - T_2$ and is stored at node X.

3) The first cycle operator, A1 circuit (T_3-T_4-T_5) makes the output Y as X^1.

4) Next, the second cycle operator, A2 circuit (T_6-T_7-T_8) results in Z to be $(X^1)^2$(which is same as X).

5) The cycle operator pair stores data in loop when WL is low ($T_{11}-T_{12}$ transmission gate is ON).

6) The circuit is in hold and stores data when WL is low and RL is low.

7) Reading starts when RL is high and the data stored is read on to line RB via transmission gate pair $T_9 - T_{10}$.

Some of the advantages of using cycle operators or buffer as storage element over STIs in ternary SRAM are:

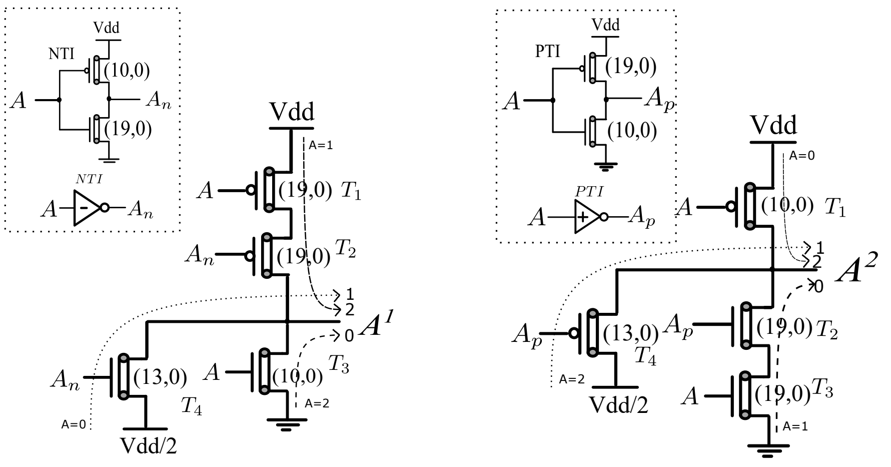
1) Two inverters are used in a STI-based buffer. So the first inverter converts ternary logic $'0'$ to logic $'2'$ before the second inverter converts it back to logic $'0'$. The cycle operator, on the other hand, converts logic $'0'$ to $'1'$ using A^1 and logic '1' to '0' using A^2. Consequently, the probability of transitioning from 0 to VDD (logic $'2'$) is greater for STI. Charging or discharging from 0 to VDD draws more current than charging or discharging from 0 to VDD/2 or VDD/2 to VDD. Thus, the cycle operator-based design uses less energy. Incorporating a transmission gate into the feedback loop to reduce the likelihood of noise signal storage in the SRAM cell yields additional power savings.

2) When a ternary buffer is used for storing a logic level in the ternary SRAM, as there is no switching here while storing data, it consumes low power. Also this design is a single stage circuit hence has low propagation delay when compared to STI based SRAM which has two stages.

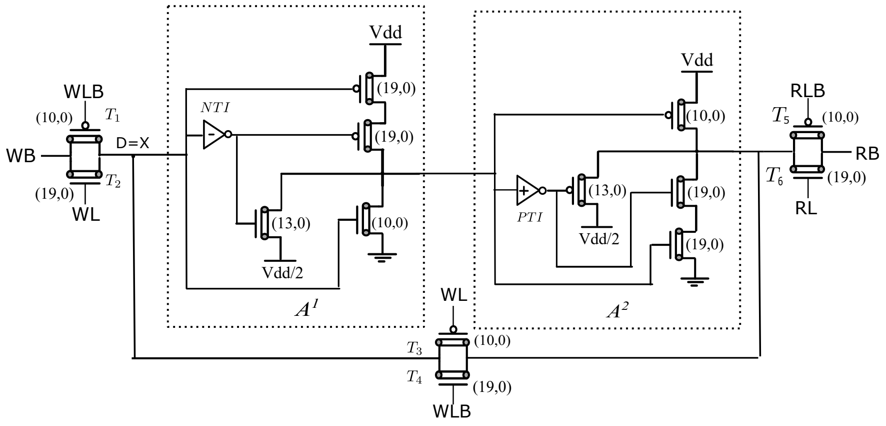
5.3 Proposed CNFET-based Ternary SRAM

5.3.1 Proposed designs for cycle operator-based Ternary SRAM

The Figure 5.2 shows the proposed designs for cycle operators and the design of a ternary SRAM using these cycle operators. The proposed cycle operators as in Figure 5.2 (a) are designed using two power supplies Vdd and Vdd/2. The cycle operators $A^1$and A^2 generate the next state and previous state of the input respectively as given in the Table 5.1.



(a) Proposed cycle operators A^1 and A^2



(b) First proposed design of cycle operator-based ternary SRAM

Figure 5.2: Proposed cycle operators and cycle operator-based ternary SRAM designs

The working of the proposed ternary SRAM in Figure 5.2 (b) is similar to the cycle operator based design in [16], described as follows. For the writing operation, when WL is high, the data D from WB line is passed to the node X via the transmission gate pair T_1,T_2. It then passes the cycle operator A^1 circuit followed by an A^2 circuit which results in the data D at the output of A^2 circuit. Both cycle operator circuits together act as a buffer. The data D is then stored in the SRAM cell by making WL low and activating the transmission gate pair T_3,T_4. For reading operation the RL is made high and the data stored in the SRAM is read out of RB via the transmission gate pair T_5,T_6. The proposed cycle operator designs used for storage in the ternary SRAM are power efficient as compared to the designs in [16].

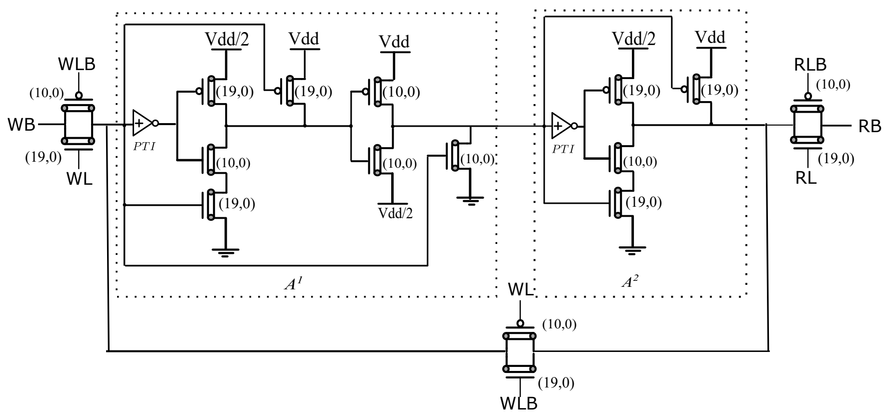


Figure 5.3: Second proposed design of cycle operator-based ternary SRAM

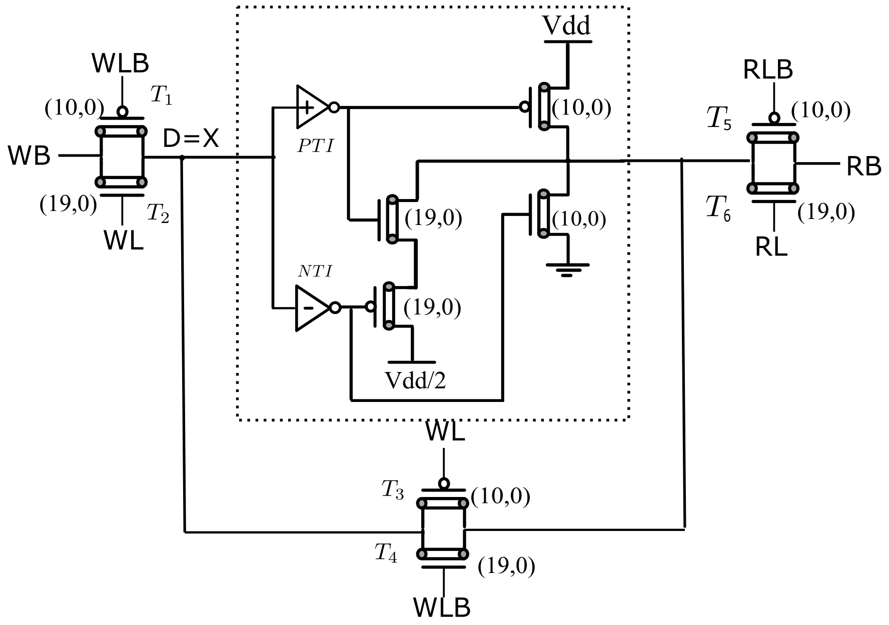


Figure 5.4: Proposed design of buffer-based ternary SRAM

Another cycle operator-based ternary SRAM is proposed in this work as shown in Figure 5.3. This design uses the cycle operators proposed in [32], which are also designed using two power supplies. The operation of the ternary SRAM proposed in Figure 5.3 is same as that of the one in Figure 5.2(b). The working of proposed cycle operator A^1in Figure 5.2 (a) is described as follows:

1) When $A = 0$, the transistors T_4 and T_1 are ON and transistors T_2 and T_3 are OFF. Hence, the output is $A^1 = 1$, as the output connects to the Vdd/2 supply.

2) When $A = 1$, the transistors T_1 and T_2 are ON and transistors T_3 and T_4 are OFF. Hence, the output is $A^1 = 2$, as the output connects to the Vdd supply.

3) When $A = 2$, the transistors T_3 and T_2 are ON and transistors T_1 and T_4 are OFF. Hence, the output is $A^1 = 0$, as the output connects to the Gnd.

The A^2 cycle operator works in a similar way.

5.3.2 Proposed design for buffer-based Ternary SRAM

A buffer-based ternary SRAM cell is proposed as shown in Figure 5.4. The ternary buffer is used as storage element in the proposed ternary SRAM. It is designed using two power supplies of Vdd and Vdd/2. This ternary buffer design is proposed in [78]. For the reading and writing operation, the working of this buffer based SRAM is similar to the working of the proposed cycle operator based SRAM. When WL is high, data D from WB is passed to node X via the transmission gate pair T_1,T_2 to the ternary buffer. The output of the ternary buffer is same as that of the input D which is stored in the cell when WL is low activating the transmission gate pair T_3,T_4. The value is read out of RB when RL is high via the transmission gate pair T_5,T_6. This buffer-based ternary SRAM design shows an improvement over the buffer-based SRAM design proposed in [16], in terms of power and number of transistors used.

5.3.3 Proposed design for a nXn Ternary SRAM

The buffer-based SRAM proposed in the previous section is found to be most efficient in terms of power and number of CNFETs used when compared to all other STI-based and cycle operator-based designs. This is because the buffer-based design is a single stage circuit that uses least number of transistors. We have used this proposed buffer-based single cell SRAM to build a $3X3$ SRAM which can store 9-trits as shown in Figure 5.5. This SRAM consists of 3 rows and each row can store a 3-trit data. The WB[2 : 0] is the 3-trit data that needs to be written in one of the rows and the row is decided by the WL that is generated using a row decoder which is of size 1 : 3 in this case as there are only 3 rows. Similarly for reading from this SRAM array, the RB[2 : 0] is the data read out from one of the rows and the row here is decided by the RL which acts as a select line for the multiplexers that are used at the output. 3,3 : 1 multiplexers are used here

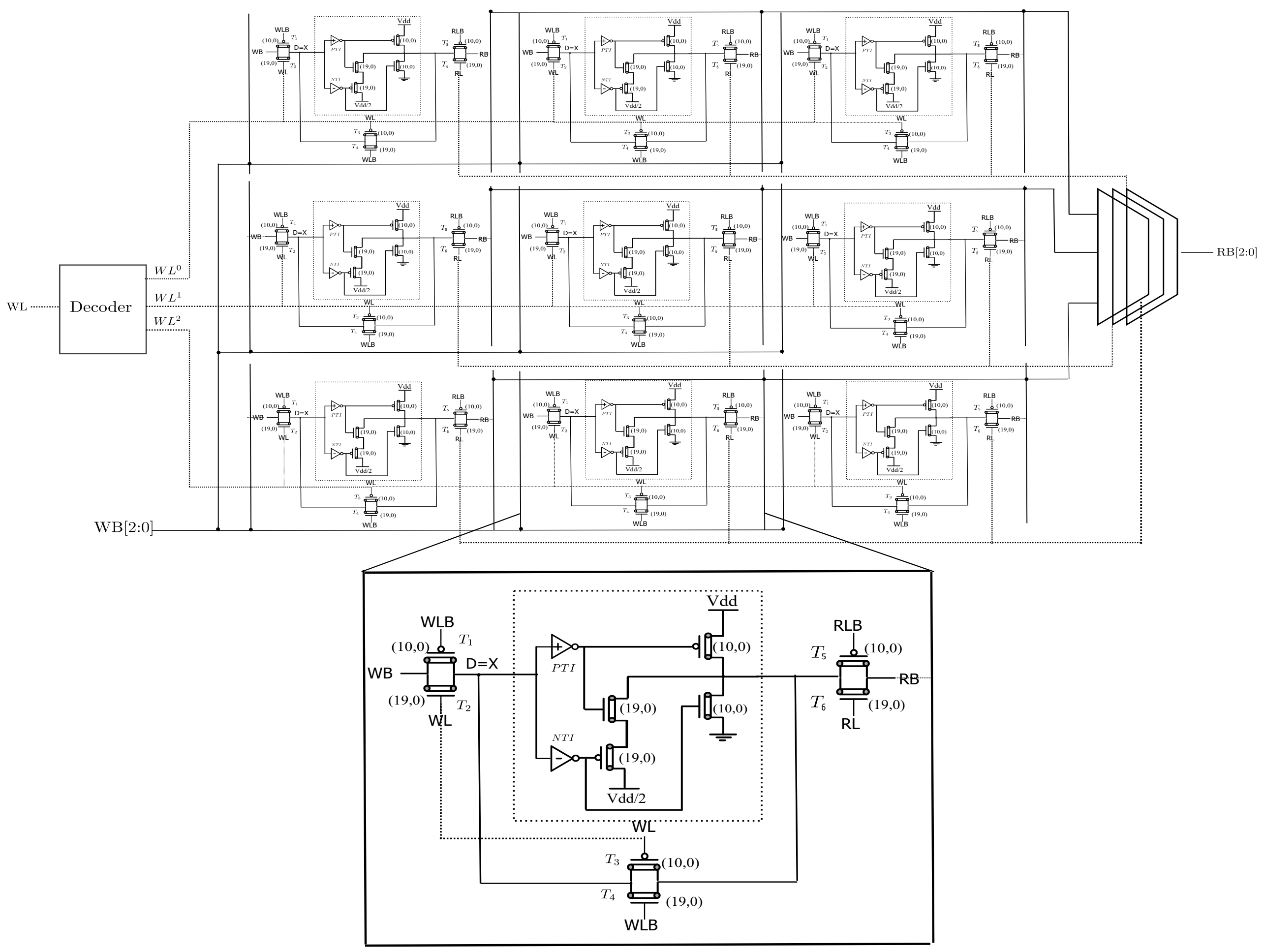


Figure 5.5: Proposed design of a $3X3$ ternary SRAM using buffer-based SRAM cell

to read the data stored from the required row. This SRAM array can be extended to store any number of digits by increasing the number of single-cell ternary SRAMs used and by increasing the size of the decoder and multiplexer as per the requirement. An extended version of this ternary SRAM of size $27X9$ and $27X3$ is used to implement the Instruction memory and the Data memory in the Chapter 6, which are used in the design of the 3-trit ternary logic processor.

5.4 Simulation Results and Discussion

Simulations were carried out using HSPICE, a circuit simulation tool by Synopsys. A standard CNFET model [21] [22] by Stanford was used in the design. The CNFETs have a default pitch value of 20 nm and were configured to have 3 CNTs. All the proposed and existing ternary SRAM designs that are compared have been given the

Table 5.2: Read and Write power(in uW) of various ternary SRAM designs

Logic	**Cycle operator-based ternary SRAM**			**Buffer-based ternary SRAM**	
	Design-1 [16]	**Proposed Design-1**	**Proposed Design-2**	**Design-2 [16]**	**Proposed Design-3**
Read Power (uW)					
'0'	1.97	0.031	0.025	0.218	0.0058
'1'	1.50	0.037	0.028	0.436	0.0124
'2'	1.78	0.026	0.023	0.219	0.0070
'Avg'	1.42	0.031	0.025	0.291	0.0084
Write Power (uW)					
'0'	2.12	1.44	0.86	0.56	0.18
'1'	2.15	0.35	0.95	0.53	0.04
'2'	1.17	0.21	0.88	0.51	0.07
'Avg'	1.81	0.67	0.90	0.53	0.10

same random input patterns and a load of 0.1fF was connected at the output of all the circuits for fair comparision. The proposed cycle operator-based designs, denoted by Proposed Design-1 and Proposed Design-2 , are compared with the cycle operator-based designs in [16], denoted by Design-1. And the proposed ternary buffer-based design, denoted by Proposed Design-3, are compared with the buffer-based design in [16], denoted by Design-2.

The Table 5.2 shows the Read and Write power consumed by the ternary SRAM designs. The Read and write power is calculated for reading and writing of logic levels 0, 1 and 2 seperately and the average power is also calculated. It is found that the Proposed Design-1 and Proposed Design-2 show an improvement of 97.8% and 98.2% in terms of average read power when compared to Design-1, respectively. Also the Proposed Design-3 shows an improvement of 97.1% in avg. read power when compared to Design-2. Similarly, the Proposed Design-1 and Proposed Design-2 show an improvement of 62.9% and 50.2% in terms of average write power when compared to Design-1, respectively. And the Proposed Design-3 shows an improvement of 81.1% in avg. write power when compared to Design-2. The Table 5.3 shows the leakage power consumed for the proposed and existing ternary SRAM designs. The leakage power is calculated while the ternary SRAM cell is storing a particular logic level either 0 or 1 or 2, and no read or write is happening. The read power, write power and static

Table 5.3: Leakage power(in uW) of various ternary SRAM cells while storing 0, 1 and 2

Static Power	**Cycle operator-based ternary SRAM**			**Buffer-based ternary SRAM**	
	Design-1 [16]	**Proposed Design-1**	**Proposed Design-2**	**Design-2 [16]**	**Proposed Design-3**
’0’	2.00	0.08	0.02	0.0007	0.003
’1’	2.00	0.04	0.03	0.84	0.024
’2’	0.71	0.001	1.42	0.001	0.002
’Avg’	1.57	0.04	0.48	0.282	0.009

Table 5.4: Read Delay(in ps) of various ternary SRAM cells

Read Data	**Cycle operator-based ternary SRAM**			**Buffer-based ternary SRAM**	
	Design-1 [16]	**Proposed Design-1**	**Proposed Design-2**	**Design-2 [16]**	**Proposed Design-3**
’0’	5.75	5.03	3.07	1.97	2.3
’1’	3.13	1.37	2.81	2.92	3.02
’2’	4.06	2.25	3.11	2.49	3.12
’Avg’	4.31	2.88	3.00	2.46	2.81

leakage power is low for the proposed designs as these use lesser number of transistors than the existing designs for ternary SRAM implementation.

The Tables 5.4 and 5.5 show the read and write delays for various ternary SRAM cells for various cases. In Table 5.4, the read delay in picosecends is for reading values of 0,1 and 2 respectively are presented along with average read delay. We observe that the Proposed Design-1 and Proposed Design-2 show an improvement of 33.1% and

Table 5.5: Write Delay(in ps) of various ternary SRAM cells

Write Data	**Cycle operator-based ternary SRAM**			**Buffer-based ternary SRAM**	
	Design-1 [16]	**Proposed Design-1**	**Proposed design-2**	**Design-2 [16]**	**Proposed Design-3**
0->1	0.76	0.68	0.84	0.81	0.71
0->2	1.29	1.61	1.51	1.65	1.44
1->0	1.07	1.56	1.05	1.05	1.02
1->2	1.52	2.57	2.24	2.01	2.03
2->0	1.28	1.54	1.32	1.12	1.15
2->1	1.54	3.20	2.72	1.94	2.17
Avg	1.24	1.86	1.61	1.42	1.42

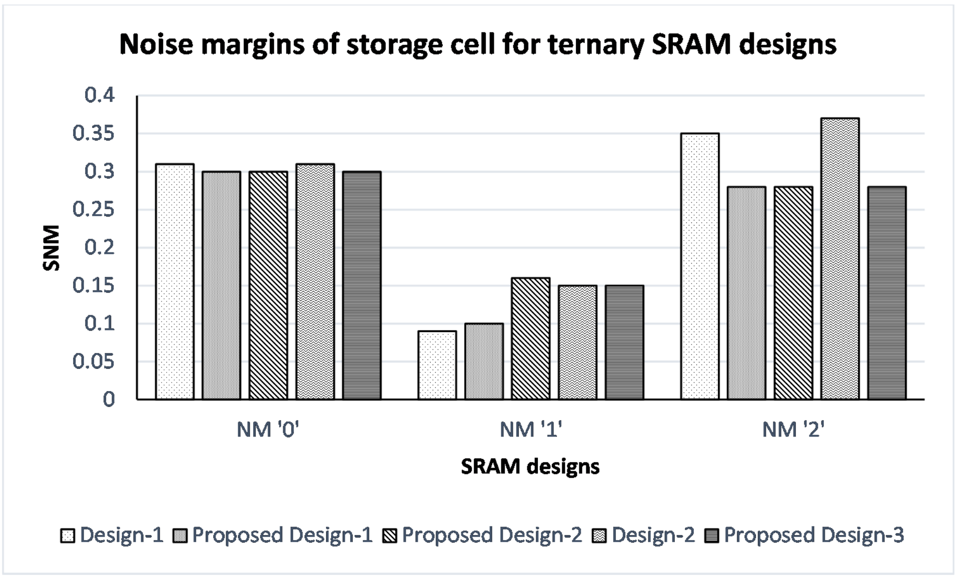


Figure 5.6: Static noise margins of various existing and proposeddesigns of ternary SRAM

30.3% respectively in read delay when compared to Design-1. In Table 5.5, the write delay is calculated for various cases. For example the case $0->1$ stands for writing a 1 into the ternary SRAM cell when it is holding a 0.

The static noise margin is calculated for both the cycle operator-based and buffer-based existing and proposed designs. The noise margin is the amount of noise that a storage cell can withstand without flipping the logic that is being stored by it. The static noise margins are calculated by inserting a voltage source at the buffer input and the at the input of the first cycle-operator in the ternary SRAM designs. The noise margins of proposed designs are found to be comparable to those of the existing designs as shown in Figure 5.6. The Figures 5.7and 5.8show the simulation waveforms for proposed cycle operator-based and ternary buffer-based ternary SRAM designs. In the waveforms, WB is the data to be written in the SRAM when WL is high, D is the data stored in the SRAM cell and RB is the data read out from the SRAM when RL is high.

In addition to this, the performance parameters for the ternary SRAM designs for variation is load, temperature and supply voltage are analysed and plotted. Here, the Design−1 and Design−2 correspond to the cycle-operator based and buffer–based ternary SRAM cells from [16](designs shown in Figure 5.1). And the Proposed Design-1

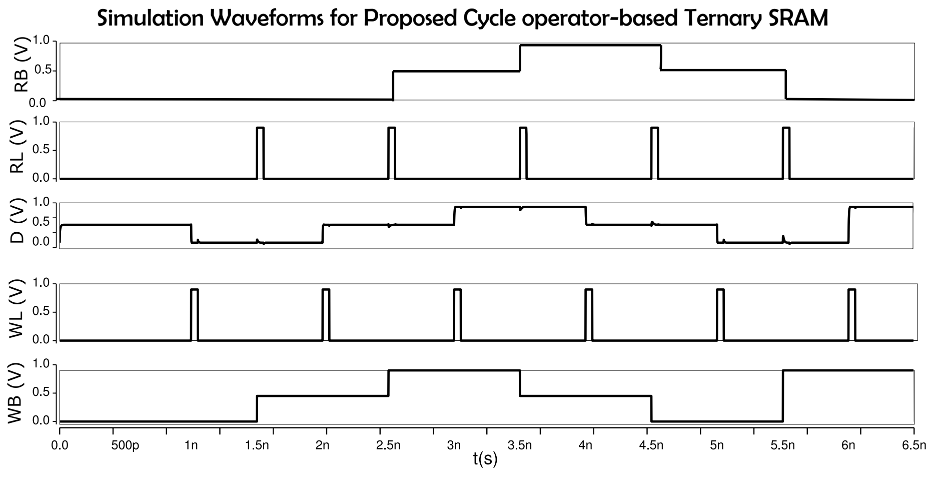


Figure 5.7: Simulation waveforms for proposed cycle operator-based ternary SRAM

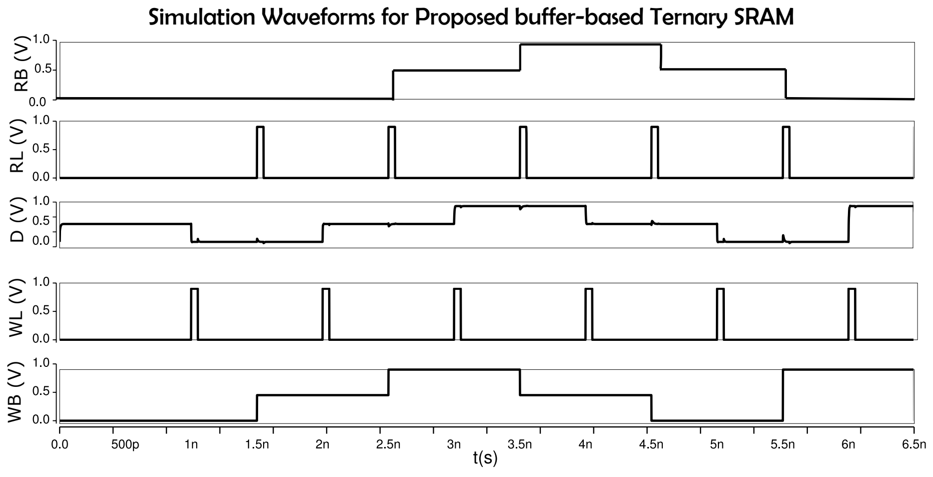
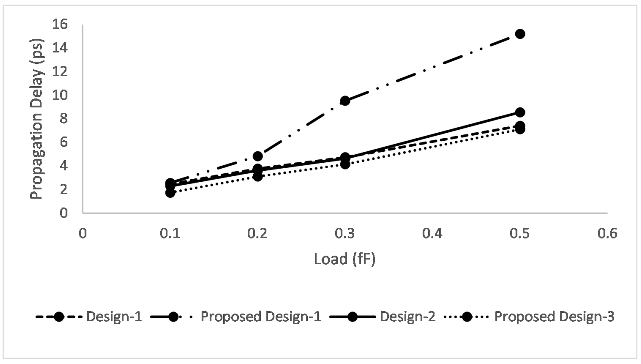


Figure 5.8: Simulation waveforms for proposed buffer-based ternary SRAM

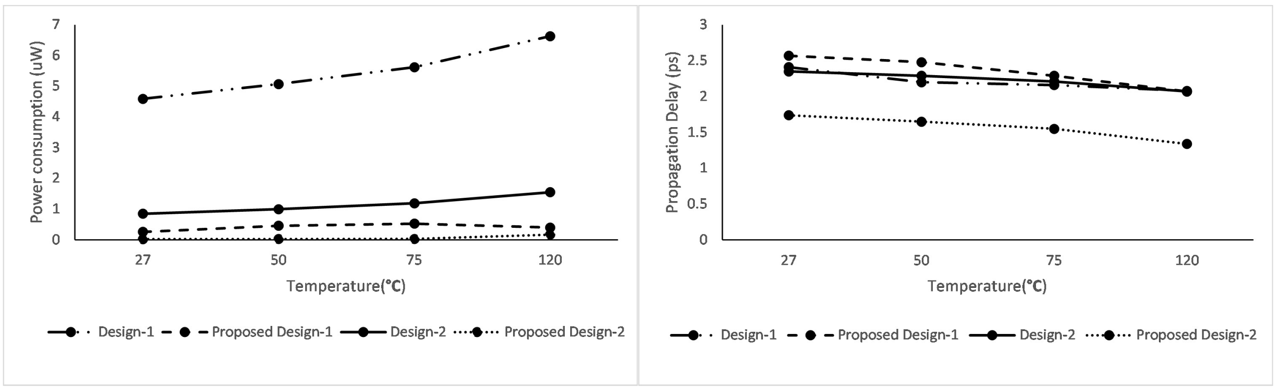
and Proposed Design-3 correspond to the cycle-operator based and buffer-based SRAM designs proposed in this chapter. The performance of different SRAM designs is calculated for the variations considering the SRAM cell is reading a logic '1' from the cell for delay calculation and it is writing a logic '1' and then reading a '1' from the cell for power calculation. So, the delay calculated in Figure 5.9 is the read delay for logic '1' and the power calculated is the power consumed for one read and one write operation of the SRAM cell.

A capacitive load of 0.1, 0.2, 0.3 and 0.5 fF is connected at the output of the SRAM cell designs that is at the 'out_rbl' line and the read propagation delay is calculated for the existing and proposed ternary SRAM cells as plotted in Figure 5.9. We can see from the graph that for varying loads the delay is more for proposed cycle operator- based



(a) Propagation delay Vs Load for ternary SRAM cell

Figure 5.9: Power and delay of proposed ternary SRAM versus existing ternary SRAM for variations in load, temperature and supply voltage

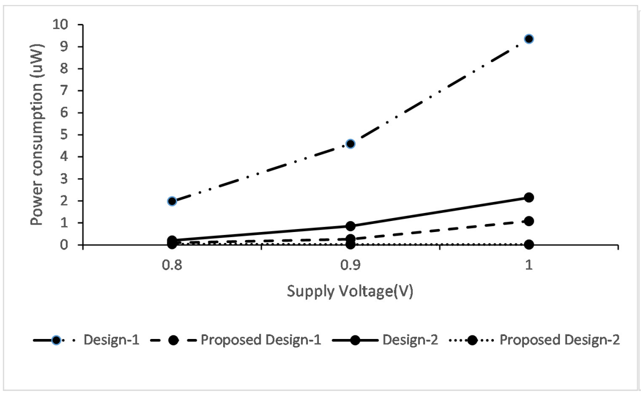


(a) Power consumption Vs Temperature (b) Propagation delay Vs Temperature

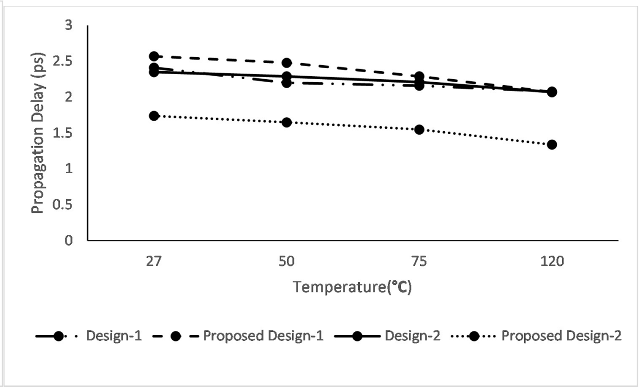
Figure 5.10: Power and delay of proposed ternary SRAM versus existing ternary SRAM for variations in temperature

designs (Proposed Design-1) as compared to the existing cycle operator-based design (Design-1). Whereas in case of buffer-based design the proposed designs have lower delay. Similarly, the Figure 5.10 show the power and delay respectively as the temperature is varied from $27 - 120^0C$. We observe that the proposed buffer-based ternary SRAM consumes the least amount of power and has the lowest delay as compared to all other designs as it is a single-stage circuit that uses least number of transistors and also uses a dual supply based design approach. The supply voltage is also varied from $0.8 - 1$ V and the corresponding power and delay are calculated for the SRAM designs as shown in Figure 5.11. Here also the proposed Design-3 of the buffer-based ternary SRAM wins both in terms of power and delay compared to all other designs.

Monte Carlo simulations are also done for the existing and the proposed ternary



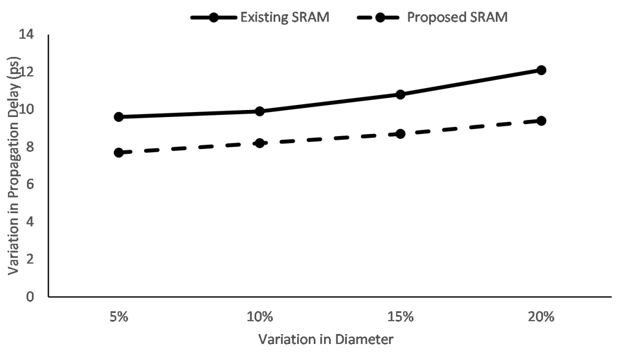
(a) Power consumption Vs Supply Voltage



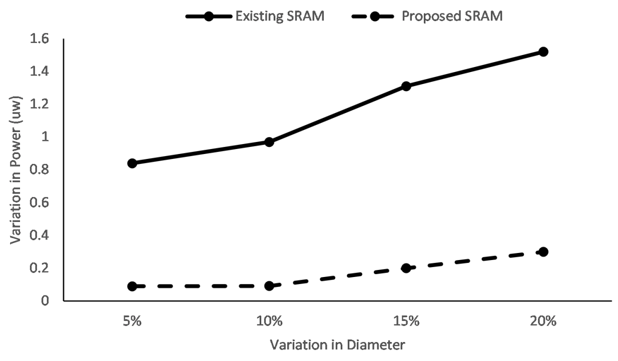
(b) Propagation delay Vs Supply Voltage

Figure 5.11: Power and delay of proposed ternary SRAM versus existing ternary SRAM for variations in temperature

SRAM designs. The variation in propagation delay and power is calculated for variations in diameter of the CNFET. the analysis is shown in Figure 5.12.



(a) Variation in Propagation delay(ps) vs diameter



(b) Variation in Power(uW) vs diameter

Figure 5.12: Effect of variation in diameter on propagation delay and power

Here the existing ternary SRAM is the buffer-based design, Design-2 and the proposed SRAM is the buffer-based proposed SRAM design, Proposed Design-3. It is observed that the effect of diameter variation has less effect on proposed designs when

Table 5.6: CNFET count for ternary SRAM designs

Design	**CNFET count**
Design-1 [16]	24
Proposed Design-1	20
Proposed Design-2	23
Design-2 [16]	20
Proposed Design-3	16

compared to existing designs. Comparision of the CNFET count for the existing and proposed ternary SRAM designs is done and presented in Table 6.7 . We observe that the proposed buffer-based ternary SRAM that is the Proposed Design-3 requires the least number of CNFETs among all the designs.

5.5 Conclusions

Three new CNFET-based designs for ternary SRAM cell are proposed in this chapter. Two designs are cycle operator-based and one design is ternary buffer-based. All three proposed designs show improvement in power consumed when compared to the ternary SRAM designs existing in literature. The read and write delays and noise margins of the proposed designs are found to be comparable to those of existing designs. HSPICE simulation results show that the proposed cycle operator based designs show an improvement of upto 98.2% in read power compared to existing cycle operator-based design. And the proposed buffer-based design shows an improvement of upto 81.1% in write power when compared to its existing counterpart.

Chapter 6

Design of CNFET-based Ternary Logic Processor

6.1 Introduction

The design of the ternary logic processor involves the integration of the following components:

1) Ternary Instruction memory - Instruction Memory is the memory that contains the programme instructions that will be executed by the processor. Typically it stores the instructions in ternary format.

2) Ternary Data memory - This is the memory used by the processor to retain the data required for calculations or operations. The data memory is composed of SRAM, which provides rapid access to data.

3) Ternary ALU - The TALU is used to perform arithematic and logic operations on the ternary operands obtained from the registers.

4) Ternary Register file - Register File is a set of high-speed storage elements used by the processor to store intermediate results, addresses, and operands. It is commonly implemented with D flip-flops and can store only a small amount of data. The register file allows the processor to quickly execute calculations and data operations.

5) Ternary Control unit - The control unit manages the flow of data and instructions

between the various processor components, such as the ALU, register file, and data memory.

The multiplexer based approach and the dual supply based approach were used to design efficient ternary logic circuits in the previous chapters. This chapter presents the design of a 3-trit ternary logic processor built using CNFETs. In Chapter 3, the design of a 2:1 multiplexer based 2-trit ALU is presented. This TALU is extended to operate on 3-trits for building a 3-trit TLP. In Chapter 4, ternary sequential circuits like D-flipflops are which are used for implementing a ternary register file for the TLP. And, the ternary SRAM cell proposed in the Chapter 5, is used for the design of an instruction memory and the data memory of the TLP.

At the onset, an instruction set architecture is presented for this TLP that consists of 14 different instructions. Next, the micro-architecture for this processor is described and the transistor level CNFET-based designs for the individual blocks of the processor like the ternary instruction fetch, the register file, the ALU, the data memory and the control unit are proposed. The entire TLP designed using CNFETs is implemented in HSPICE to measure it performance and the functionality of the proposed processor is verified with the help of standard programs.

The Section 6.2 described the work related to the ternary processor that is present in the existing literature. Section 6.3 presents the design of proposed TLP and the proposed TLP's blocks. The Simulation results are discussed and presented in Section 6.4 and conclucions are drawn in Section 6.5.

6.2 Related Work

The research work related to the ternary logic processor implementation is published in [18] and [19]. In [18], the design and verification frameworks for developing a fully-functional emerging ternary processor are described. It presents a top-level ternary microprocessor based on a 9-trit instruction-set architecture with 24 custom ternary instructions. The proposed software-level framework provides an efficient way to convert existing binary programs to ternary codes, while the hardware-level framework offers a cycle-accurate simulator and a technology mapper for quantitative evaluations of the pipelined ART-9 (Advanced RISC-based Ternary) architecture for arbitrary design technology. As a case study, the proposed ART-9 core achieves a processing efficiency of 57.8 DMIPs/W and 3.06×10^6 DMIPs/W when using FPGA-level ternary-logic emulations and CNFET-based ternary gates, respectively. Although, this study is significant as it provides a comprehensive and quantitative evaluation of a fully-functional ternary microprocessor, it does not describe the transistor-level implementation of the ternary processor, that is described in the proposed section of this paper.

Another paper [19], proposes a novel approach to designing and implementing an efficient instruction set for a ternary processor using VHDL. The paper considers 21 instructions, including various addressing modes such as register, direct, and immediate modes. This research is significant because it addresses the need for efficient instruction sets for ternary processors. The use of VHDL in the design and implementation of the instruction set provides a reliable and standardized way of designing digital circuits. The consideration of various addressing modes provides flexibility in programming, while the simulation results provide evidence of the efficiency of the proposed instruction set. Overall, this research contributes to the development of more efficient and flexible ternary processors. Further enhancements to the designed instruction set may lead to even better performance and capabilities for ternary processors.

6.3 Proposed Design of a Ternary Logic Processor

This section presents the design of a CNFET-based Ternary Logic Processor (TLP). This TLP is a single-cycle processor that can handle 3-trit data and can execute 14 different instructions. A digit in ternary is called as a trit. The address size is also 3-trits and each instruction is 9-trits long. There are 9 different registers, each register is of size 3-trits. The TLP has an instruction memory and a data memory that are of sizes $27X9$ and $27X3$, respectively.

Firstly, an ISA is designed for the ternary logic processor as described in Section 6.3.1. Next, the Section 6.3.2 presents the Micro-architecture of the proposed ternary

Table 6.1: The proposed instructions for the Ternary Logic Processor

S.No.	**Mnemonic**	**Type**	**Opcode ($I_2 I_1 I_0$)**	**Example**
1	tadd	R	000	tadd t0,t1,t2
2	tsub	R	100	tsub t0,t1,t2
3	tmul	R	200	tmul t0,t1,t2
4	tand	R	001	tand t0,t1,t2
5	tor	R	101	tor t0,t1,t2
6	txor	R	201	txor t0,t1,t2
7	tnand	R	002	tnand t0,t1,t2
8	tnor	R	102	tnor t0,t1,t2
9	taddi	I	$x10$	taddi t0,t1,3
10	tlw	I	$x11$	tlw t0, 4(t1)
11	tsw	S	$x12$	tsw t1, 6(t2)
12	tbgt	B	$x20$	tbgt t1,t2,L1
13	tblt	B	$x21$	tblt t1,t2,L1
14	tbeq	B	$x22$	tbeq t1,t2,L1

logic processor. The fundamental blocks required for the Ternary logic Processor are the Ternary Instruction Fetch unit (TIF), the Ternary Register File (TRF), the Ternary ALU (TALU), the Ternary Data Memory (TDM) and a Ternary Control Unit (TCU). Transistor level designs using CNFETs for each of these blocks are presented in Sections 6.3.3-6.3.7.

6.3.1 Proposed Instruction Set Architecture

An Instruction Set Architecture (ISA) is proposed for the ternary logic processor which consists of various R-type (Register-type), I-type(Immediate-type), S-type (Store-type) and B-type(Branch-type) of instructions. These instructions are of 9-trit length ($I_0 - I_8$) and the instruction format is defined in the Figure 6.1. The first three trits of the instruction ($I_0 - I_2$) define the opcode. The $'rs1'$ and $'rs2'$ are the source registers and $'rd'$ is the destination register in the instruction. The 'imm' stands for the 3-trit immediate value and the 'Label' in the branch instructions signifies the address of the location to which it should branch. The various instructions and their corresponding types, mnemonics and opcodes are presented in Table 6.1. 14 different instructions are defined as part of this ISA.

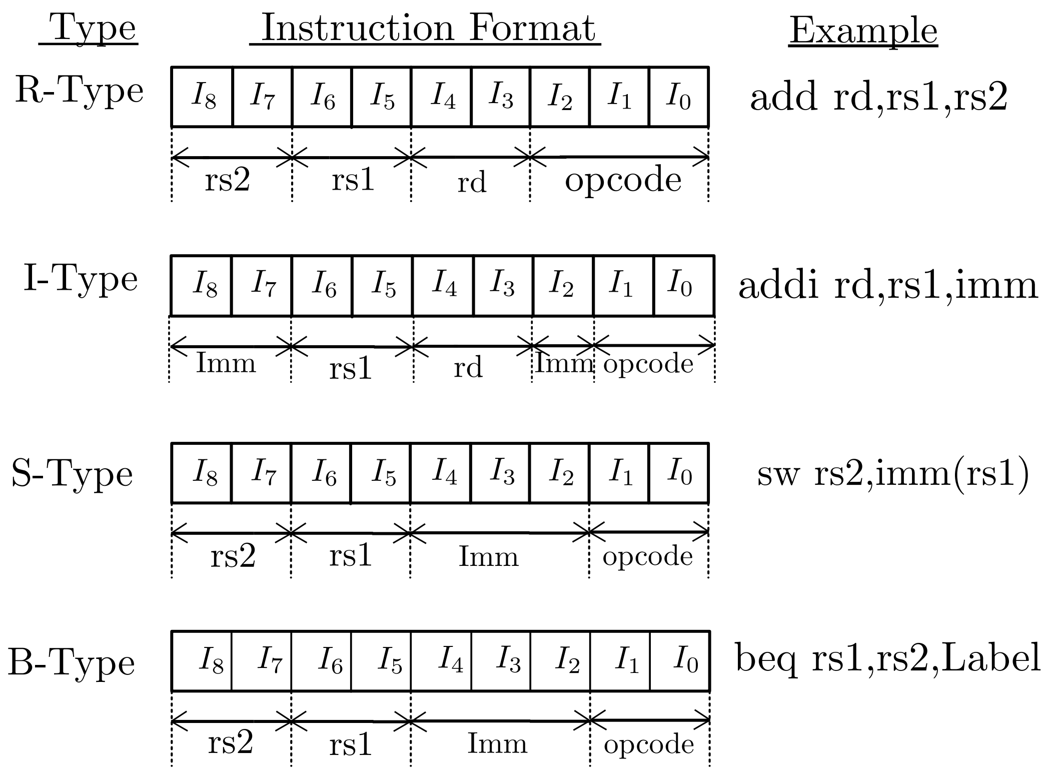


Figure 6.1: Proposed 9-trit Instructions Format

The R-type instructions perform arithmetic and logic operations like addition, subtraction, AND, and OR on operands in two registers and place the result in the third register. An example of the R-type instruction is tadd t0,t1,t2, here, addition is performed on the data in registers t1 and t2 and the result is placed in t0 register. The I-type instruction performs operations on immediate data. An example is taddi t0,t1, 2, here, a constant value 2 is added to data in t1 and the result is placed in register t0. The 'tlw' and 'tsw' are the load and store instructions. The 'tlw' instruction loads data from memory into a register and the 'tsw' instruction stores data from register into the memory. The B-type instructions are tbgt, tblt and tbeq which stand for branch-if-greater-than, branch-if-less-than, and branch-if-equal. Here, the data in two registers are compared and the instruction branches to a particular location if the compared value corresponds to a less than, a greater than, or equal to.

Remark 1 : The proposed ISA defines a total of 14 instructions in Table 6.1, out of which the first 8 instructions are of R-type, next 2 instructions 'taddi' and 'tlw' are categorized as I-type as they have the same instruction format, 1 instruction is of S -type the 'tsw' and 3 instructions are of B-type.

6.3.2 Proposed Micro-Architecture

The architecture of the proposed Ternary logic Processor is shown in Figure 6.2 . It consists of the typical Instruction Fetch, Decode, Execute, Memory, and Write-back stages. The working of the TLP is explained as follows:

1) The first stage in the TLP is Instruction 'Fetch'. Initially, a set of instructions is loaded into the Ternary Instruction Memory (TIM) with the help of a reset signal. A Program Counter (PC) register, which contains a 3-trit address, is used to point to the current instruction that is to be executed in the TIM. This PC address is given as input to TIM. The output of the TIM is the 9-trit instruction stored, that is stored at the address in the PC. An adder is also used in this stage to increment the PC to point to the next instruction.

2) Next stage is the 'Decode' stage. In this stage, the instruction fetched from the TIM, is decoded to extract the information such as the opcode, the source and destination operand registers, the immediate data. Ternary Register File (TRF) module takes the source register extracted from the instruction code as input and gives the data present in that corresponding register as output. This data is further given as inputs to the Ternary Arithmetic and Logic Unit (TALU). The TRF is made up of 9, 3-trit registers. Whenever the signal regwrite is active, data can be written into the registers in TRF.

3) The 'Execute' stage of TLP operates on the data taken from the registers using a TALU. This 3-trit TALU performs arithmetic and logic operations on the operands obtained from the TRF and gives a 3-trit result as output. This ALU result is either written back to the TRF or goes as input to the Ternary Data Memory (TDM) depending on the type of instruction.

4) The next stage of the processor is the 'Memory' stage. In this stage, we can access the data memory either by reading from it or writing into it. This TDM stores 3-trit data at 27 different locations. Data is either written into or read out of the data memory depending on the type of instructions and the 'memread' and 'memwrite' control signals. Finally, the data read from the TDM or data from the TALU is written

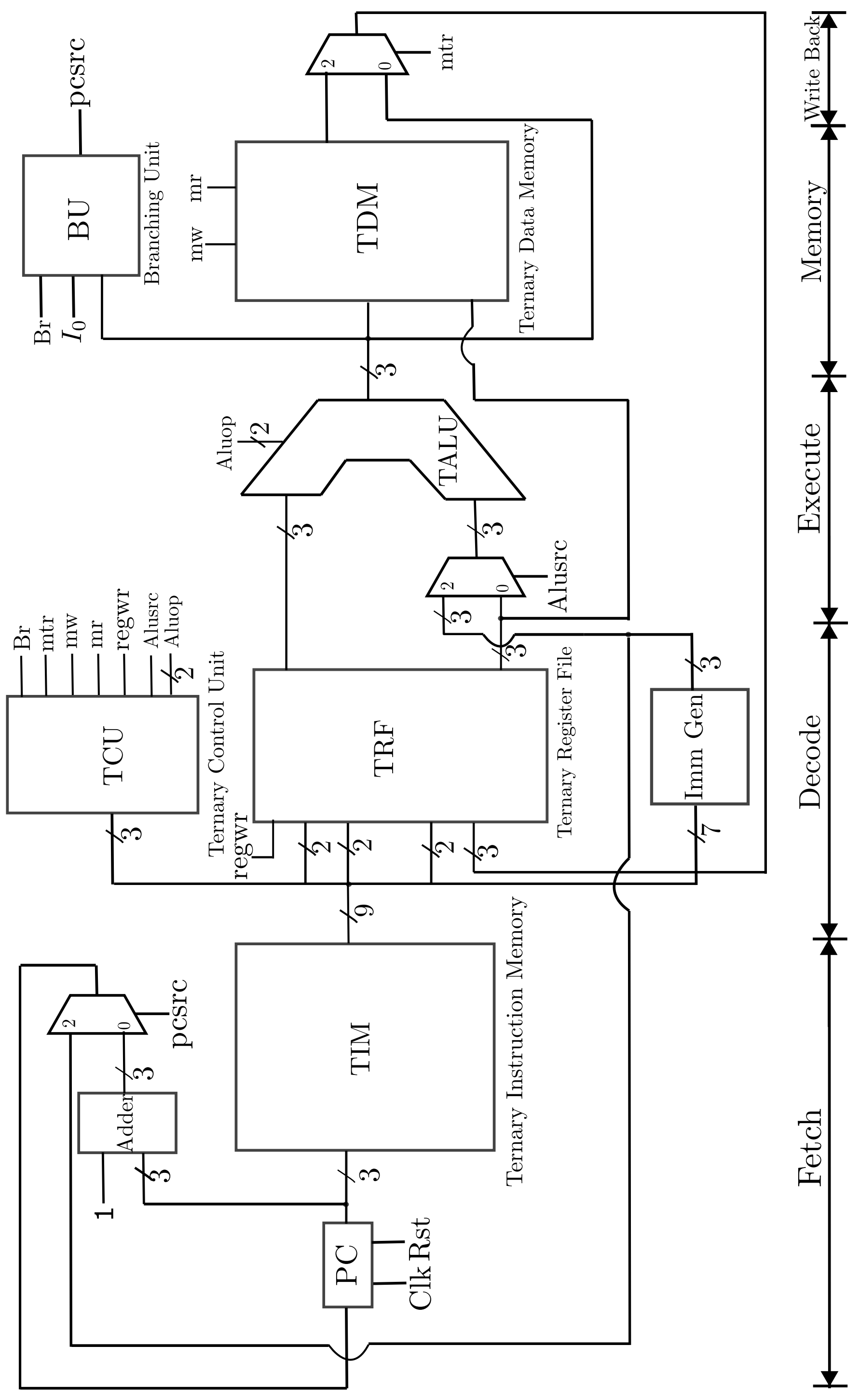


Figure 6.2: Proposed Architecture for a CNFET-based Ternary Logic Processor

back into the TRF when required, in the 'Write Back' stage of the TLP.

5) A Ternary Control Unit (TCU) controls the operation of the entire processor. It takes a 3-trit opcode as input and gives different control signals as output that are required at various stages in the datapath of the processor.

6) In addition to the above blocks, the processor design also requires an Immediate generation block, a branching unit, and a few multiplexers as shown in Figure 6.2. The Immediate generate block generates the 3-trit immediate value by taking inputs from the 9-trit instruction. This immediate value is given as the second input to the TALU for instructions that are not R-type. A branching unit is used to generate a control signal that can modify the PC value to point to the branch location mentioned in the branch type instructions. In branch type instructions, the ALU compares the two data values and outputs it to the branching unit. Branching unit also takes the opcode and the 'branch' control signal of the TCU as inputs. It generates 'PCsrc' signal as output. This 'PCsrc' is responsible for branching to the given location by making the PC point to the branch location in the TIM.

As shown in Figure 6.2, a set of 3, 2 : 1 multiplexers are also required at three different places in the proposed processor design.

Remark 2: The first set of multiplexers is used at the input of the second operand to the TALU. The control signal 'Alusrc' is used as the select line for these multiplexers and depending on if the instruction being executed is of R-type or not, the 'Alusrc' selects the operand read directly from the output of the TRF or from the Immediate generate block.

Remark 3: The second set of multiplexers is used at the output of the TDM. This uses the 'mtr' control signal as the select line for selecting either the data to be written back to the TRF either from the TDM or from the TALU output depending on the instruction being executed.

Remark 4: The third set of multiplexers is used at the output of the adder in the TIF unit. This uses the PCsrc signal from the branching unit as the select line to choose to branch to a particular location or not, depending on the type of instruction.

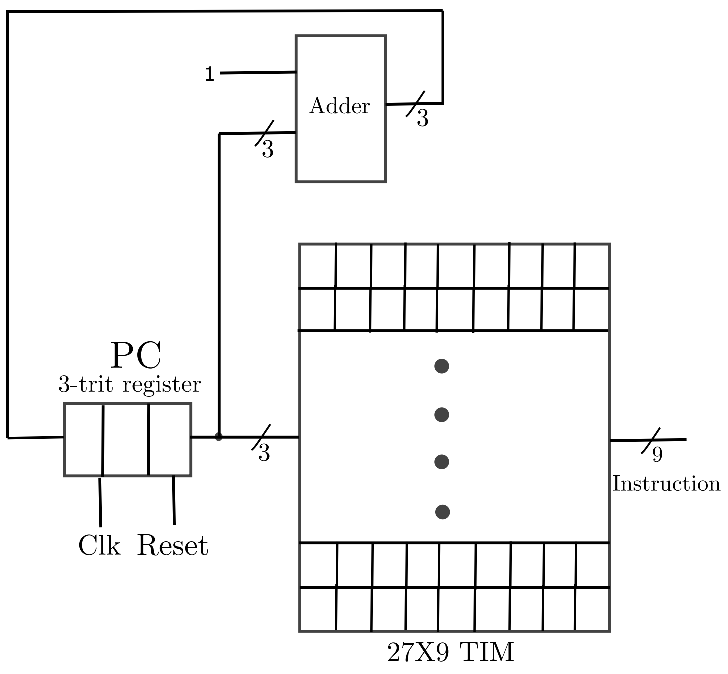


Figure 6.3: Proposed design of Ternary Instruction Fetch Unit (TIF)

6.3.3 Proposed Design of Ternary Instruction Fetch Unit

The Instruction Fetch Unit is the first block of the processor which is responsible for fetching an instruction for the processor. Typically, it consists of a instruction memory that stores the instructions, a program counter register that stores the address of the current instruction, and an adder that is used to calculate the address of the next instruction. The implementation of a Ternary Instruction Fetch (TIF) Unit is shown in Figure 6.3 .

The TIF consists of a Program Counter (PC) register, which is of 3-trit length and is designed using three ternary D-flipflops. The D-flipflop used here is a ternary buffer–STI-based flipflop as presented in Section 6.3.4. This register stores the address of the current instruction being fetched from the instruction memory by the processor. A ternary adder is used to go to the address location of the next instruction to be fetched. This adder adds a constant value of 1 to the current PC value and sends the result back to the PC, at every positive edge of the clock to go to the next location.

The clock and reset are given as inputs to both the PC register and the instruction memory. The PC is initialized to zero when the reset is HIGH and when the reset is LOW the PC increments for every clock edge. The Ternary Instruction Memory (TIM)

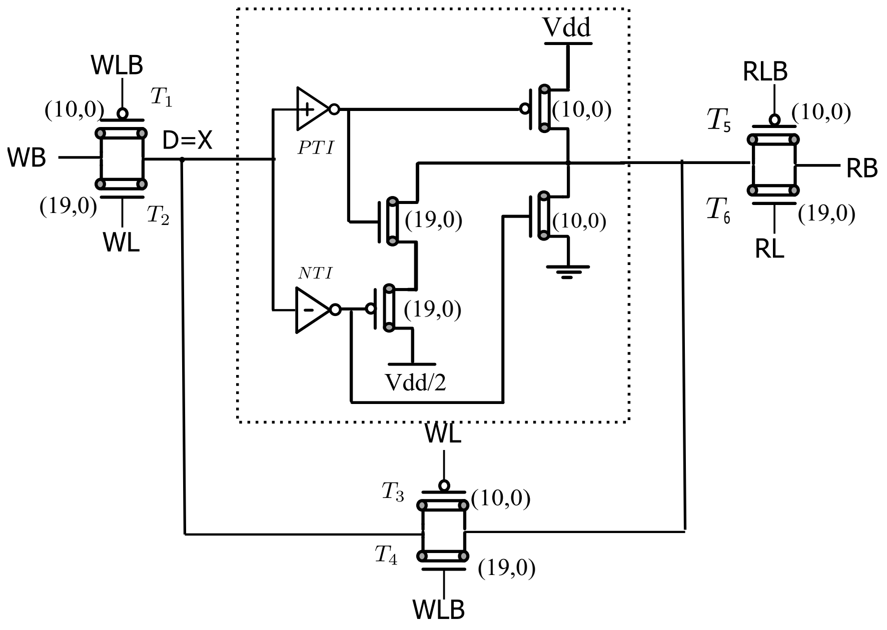


Figure 6.4: Design of ternary SRAM cell used in Ternary Instruction Memory [14]

stores the set of instructions that need to be executed by the processor. The TIM takes the current value of the PC as an input which is a 3-trit address and gives the 9-trit instruction stored at that location as output (which is also the output of the TIF) at every positive edge of the clock.

Remark 5: The TIM is initialized with a set of instructions when the reset is HIGH and it stores these instructions. This TIM is designed to store 9-trit long instructions at each address location. The address length is 3-trits so there are a total of 27 locations in the TIM where each location stores a 9-trit value. Each digit of the 9-trit instruction is stored in a single-cell SRAM as shown in Figure 6.4 . Hence the entire TIM is a $27X9$ ternary SRAM.

6.3.4 Proposed Design of Ternary Register File

The instruction code which is the output of the TIF unit is used to give inputs to the ternary register file (TRF), as shown in the Figure 6.2. The proposed TRF, shown in Figure 6.5 , consists of 9 registers (t0-t8) each 3-trit long. Each register is designed using three ternary D-flipflops. The ternary D-flipflop used to design the register is similar to the ternary buffer-STI based D-flipflop proposed in [78]. The transistor level designs of the ternary buffer and STI used in this ternary D-flipflop are also shown in

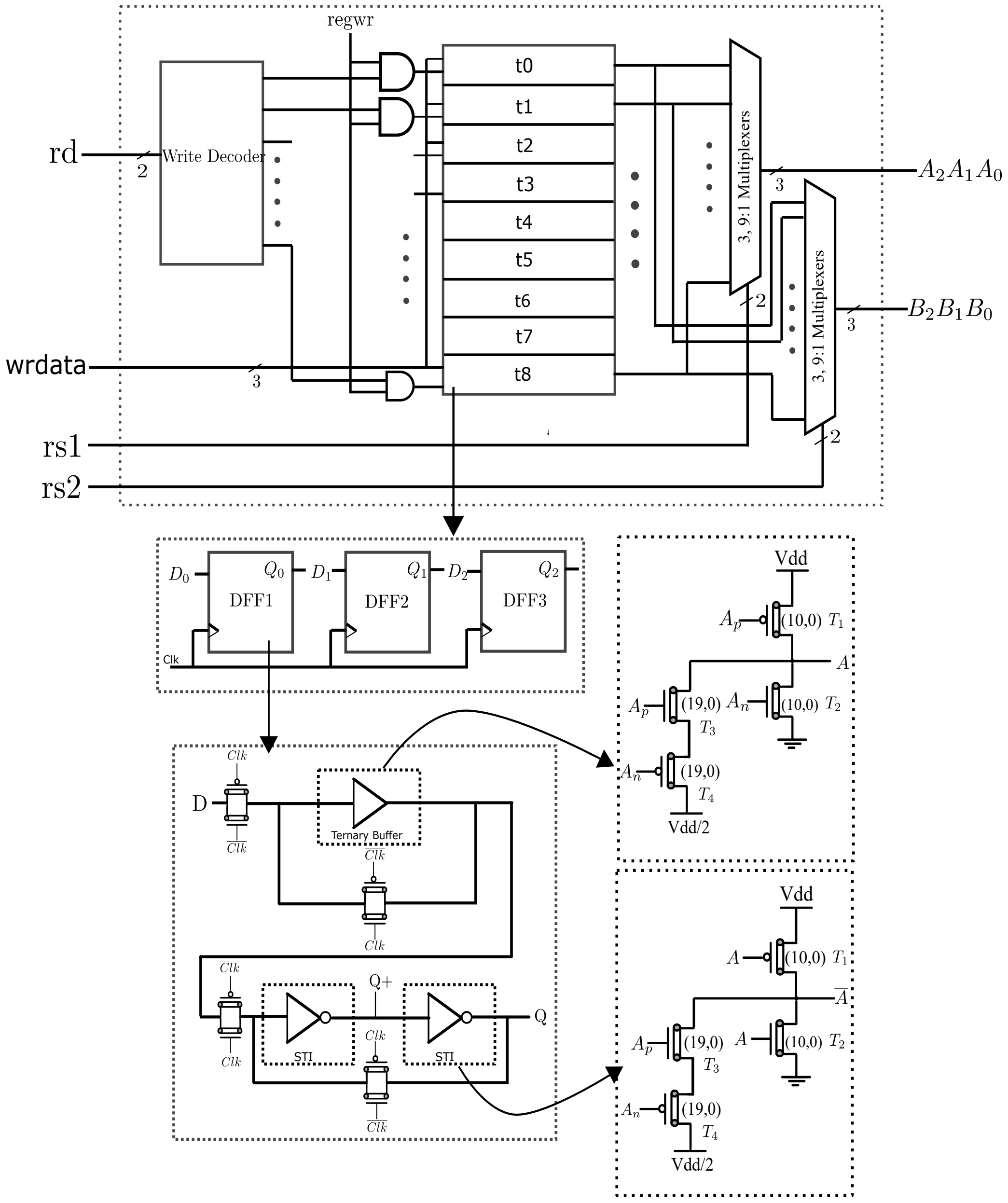


Figure 6.5: Proposed design of the Ternary Register File (TRF)

Figure 6.5.

The inputs to the TRF are read register numbers (rs1,rs2), write register number (rd) and write data (wrdata). The registers in the TRF are initialised to zero by applying a reset signal that is HIGH initially and is LOW later. Since there are 9 registers, each register can be represented by a $2-$trit number. So, the read register numbers rs1 and rs2, 2-trit long are taken from the instruction code. Also the destination register number rd is 2-trits long. The 3-trit data that is to be written in the registers indicated by 'wrdata'. The data in the input register numbers is read out as the output of the TRF.

A $2 : 9$ write decoder is used to write data into the registers in the TRF. The CNFET based design of the write decoder used here is similar to the one in [79]. Data is written into the registers only when the regwrite signal is HIGH. This is done by using an enable for each register. The value from the output of the write decoder and the regwrite signal are given to AND gates and this output is given to the Enable signal of the register as shown in Figure 6.5. Also to read out the data from a specific register $3, 9 : 1$ multipliexers are used. Hence, a set of two $3, 9 : 1$ multiplexers are used to read out data from two registers of the TRF. Each 9 :1 multipliexer is designed using four $3 : 1$ multiplexers as in [12], shown in Figure 6.6 .

6.3.5 Proposed Design of a Ternary Arithmetic Logic Unit

The Ternary Arithematic and Logic Unit (TALU) is used to perform arithmetic and logic operations. The proposed 3-trit TALU implemented for the ternary processor is shown in Figure 6.7 . The architecture of the proposed 3-trit TALU is similar to the $2 : 1$ multiplexer based 2-trit TALU proposed in [12], that is, the 2-trit TALU in [12] is extended to a 3-trit TALU here. All the functional blocks like the adder-subtractor, multiplier, comparator and logic gates are designed using 2:1 multiplexer based design approach and can handle 3-trit data.

This TALU takes two 3-trit inputs $A_0A_1A_2$ and $B_0B_1B_2$ and gives a 3-trit output $T_2T_1T_0$ as shown in Figure 6.7. The function select block takes inputs S_0S_1 from the

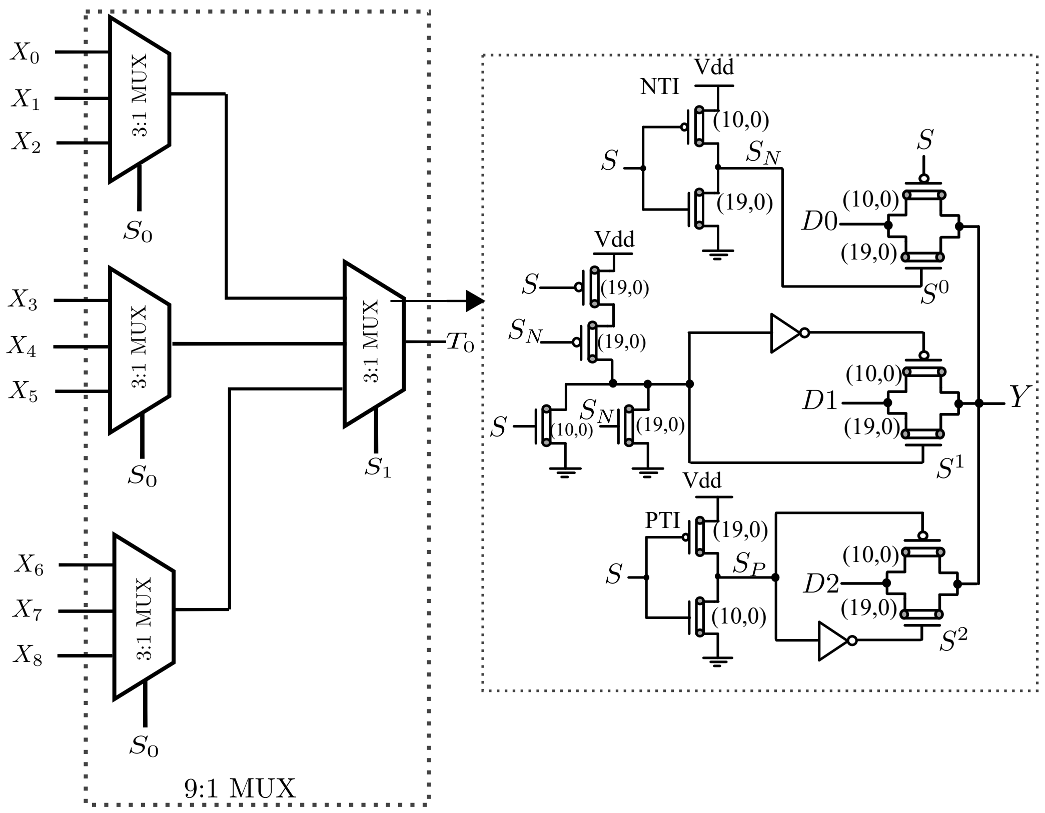


Figure 6.6: Design of 9:1 multiplexer using 3:1 multiplexers [12]

ternary control unit and is responsible for enabling the desired TALU function via the transmission gates. Six transmission gates are present in each TG block that connect the inputs to the functional blocks like adder-subtractor, multiplier, comparator, etc for performing the arithmetic and logic operations. At the output three $9:1$ multiplexers are used, one for each TALU output which chooses from one of the functional block outputs that would go as output to $T_2T_1T_0$ based on the S_0S_1 select lines.

Remark 6 : The 2:1 multiplexers used for the design of these functional modules are either PTI mux or NTI mux shown in Figure 6.8 . These 2:1 multiplexers are same as the ones used in [12].

6.3.6 Proposed Design of Ternary Data Memory

The Ternary Data Memory (TDM) is used to store the ternary 3-trit data. Each location of the TDM is 3-trit long and can be addressed using 3-trits, hence there are total 27 possible locations for the proposed memory. The TDM is implemented using single cell SRAMs. Each single cell ternary SRAM stores 1-trit data. Therefore the

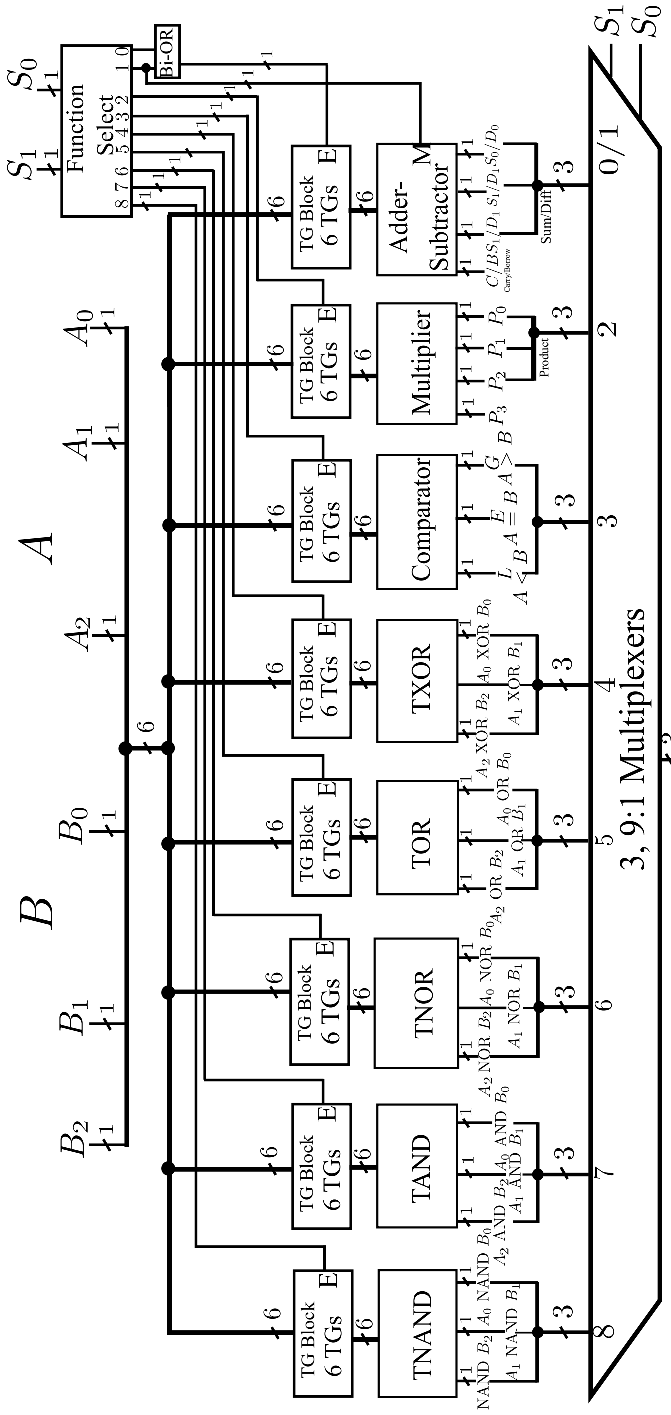


Figure 6.7: Design of Proposed 3-trit Ternary Arithmetic and Logic Unit (TALU)

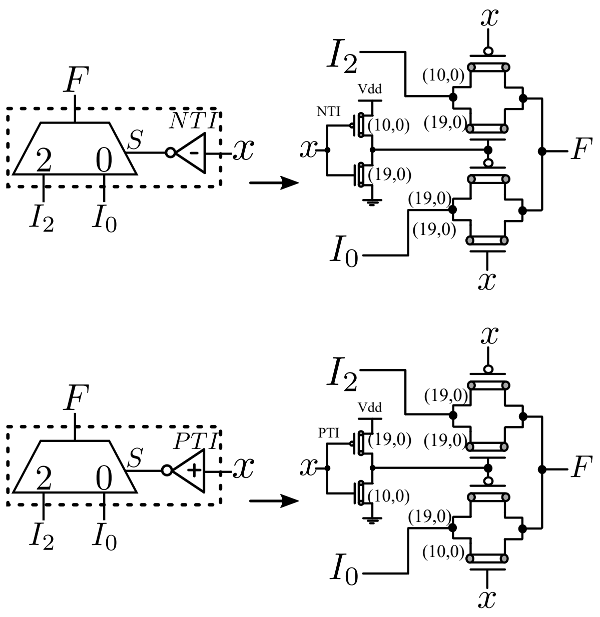


Figure 6.8: Design of 2:1 multiplexers used for the TALU [31]

TDM is implemented as a $27X3$ ternary SRAM as shown in Figure 6.9. The TDM is used in the datapath of the TLP whenever a load or store instruction is being executed. The address of the TDM that needs to be written or read is provided by the TALU. Each location of the TDM is denoted by a word which is 3-trit long.

The inputs to the word, shown in Figure 6.9 , are the 3-trit address location denoted by wbl[2:0], the data to be written denoted by wdata[2:0], the memread (mr) and memwrite (mw) control signals coming from the control unit. In addition to these inputs there are inputs init[2:0] , which is the initial data that needs to be written into the word, and the reset, which when HIGH initialises the TDM, and writes the wbl into it, otherwise. This is done by using a set of 3, $2:1$ muxes for which the reset is the select line. The data read out from the TDM is denoted by rbl[2:0]. The proposed TDM mainly consists of a $3:27$ write decoder, 27 $3-$trit words and multiplexers. The decoder takes $3-$trit address location as input and generates 27 decoder outputs, each of which correspond to one location in TDM. There are $27,3$-trit words where each word is implemented using 3 buffer-based single cell ternary SRAMs. The design of the single cell SRAM used here is similar to the SRAM design in [14]. Each location in the TDM is written into when memwrite (mw) is HIGH and the value from the particular location is read out when memread(mr) is HIGH.

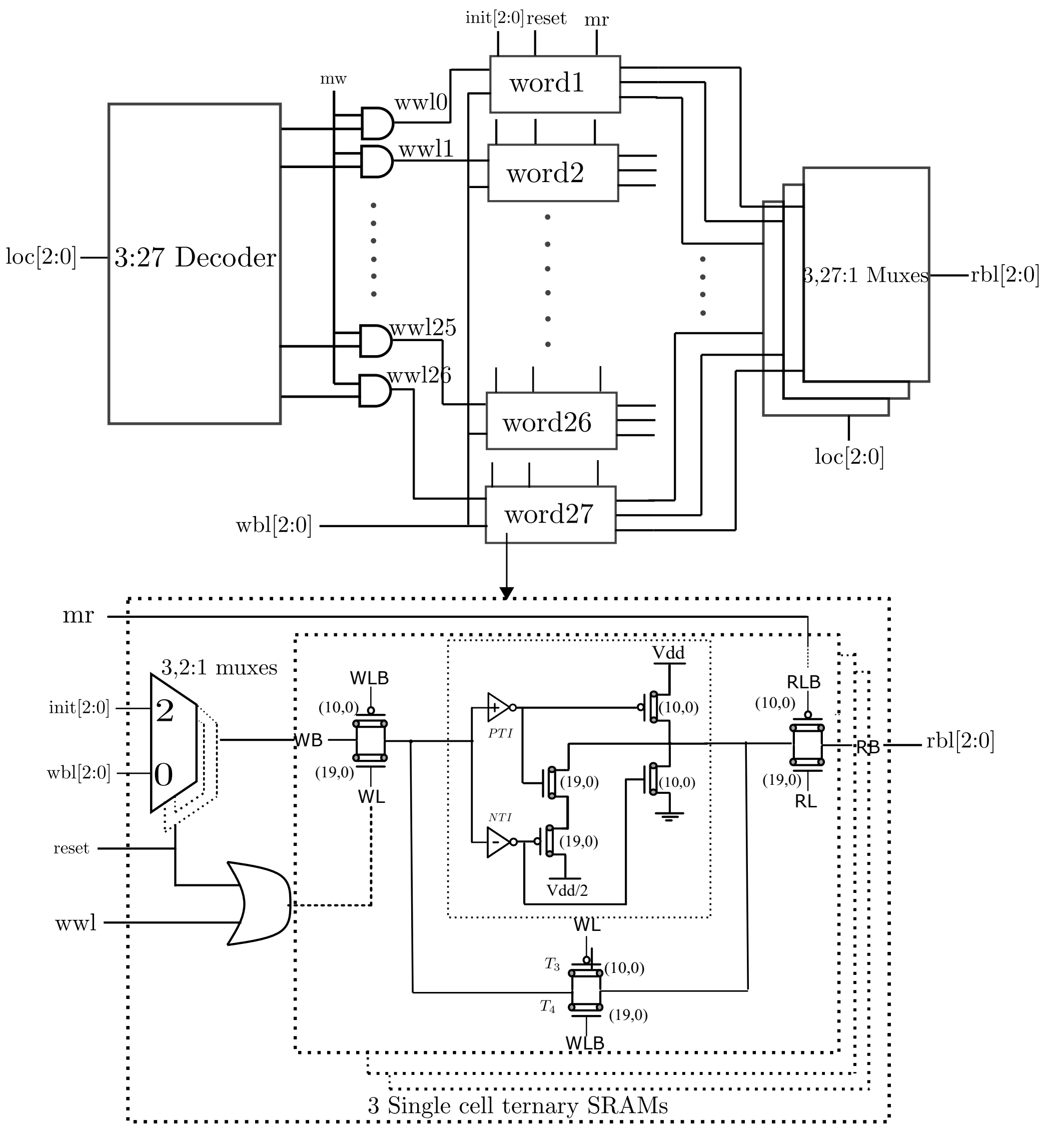


Figure 6.9: Proposed $27X3$ Ternary Data Memory built using 3-trit words

6.3.7 Proposed Design of Ternary Control Unit, Branching unit and Immediate Generation Unit

In addition to the above blocks, a Ternary Control Unit (TCU), a Branching Unit (BU) and a Immediate Generation Unit (IGU) are also required for the complete implementation of a TLP.

The TCU is responsible for the control path of the TLP. It takes the opcode of 3-trits as input coming from the TIM and gives various control signals as output. The truth table for the proposed TCU is as shown in Table 6.2. There are 7 different control signals that are generated by the TCU, namely,'Br', 'mtr', 'mr', 'mw', 'regwr',

Table 6.2: Truth Table for control signals generation in TCU

Instruction	I_2	I_1	I_0	**Br**	**mtr**	**mr**	**mw**	**regwr**	op_1	op_0	**Alusrc**
tadd	0	0	0	0	0	0	0	2	0	0	0
tsub	1	0	0	0	0	0	0	2	0	1	0
tmul	2	0	0	0	0	0	0	2	0	2	0
tand	0	0	1	0	0	0	0	2	2	1	0
tor	1	0	1	0	0	0	0	2	1	2	0
txor	2	0	1	0	0	0	0	2	1	1	0
tnand	0	0	2	0	0	0	0	2	2	2	0
tnor	1	0	2	0	0	0	0	2	2	0	0
taddi	x	1	0	0	0	0	0	2	0	0	2
tlw	x	1	1	0	2	2	0	2	0	0	2
tsw	x	1	2	0	0	0	2	0	0	0	2
tbgt	x	2	0	2	0	0	0	0	1	0	0
tblt	x	2	1	2	0	0	0	0	1	0	0
tbeq	x	2	2	2	0	0	0	0	1	0	0

'op1' and 'op0', and 'Alusrc'. 'Br' stands for the branch control signal which is HIGH only for the branch type of instructions (tbgt,tblt and tbeq). 'mtr' is the memory to register control signal, that is HIGH when the data needs to be written from the memory to register in case of the load word (tlw) instruction. The 'mr' and 'mw' stand for memread and memwrite control signals that are HIGH for load (tlw) and store (tsw) instructions, respectively. The 'regwr' stands for register write that is HIGH when data is to be written in the register in TRF for all R-type, I-type and load word instructions. The op_0 and op_1 signals are the select signals for the TALU which decide the function the TALU should implement. The 'Alusrc' signal is HIGH for the Immediate, load and store instructions (taddi, tlw, tsw) since for these instructions the second operand to the TALU is the immediate data generated by the IGU. The TCU is completely designed using 2:1 multiplexers as shown in Figure 6.10.

The BU is implemented as shown in Figure 6.11. It takes the input as the 3-trit output of the ternary comparator in the TALU and the Br control signal to generate the PCsrc as the output. The 'PCsrc' is used to branch to the required location. The BU is implemented using a 3 : 1 multiplexer and an AND gate.

The IGU is as shown in Figure 6.12. This unit is used for generating the immediate data from the instruction code. The immediate data is of size 3-trits (Imm[2:0]). The

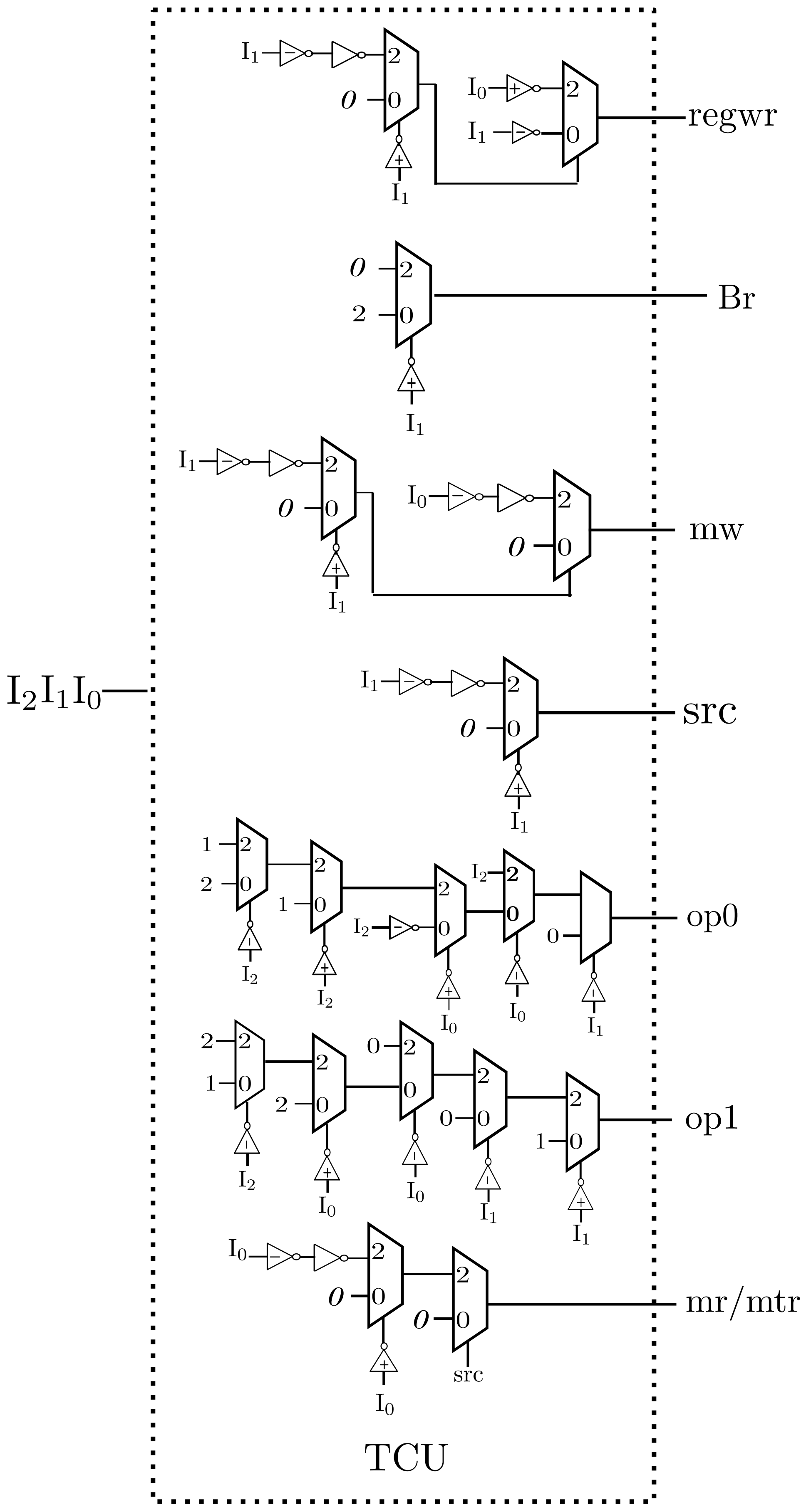


Figure 6.10: Proposed Design of Ternary Control Unit (TCU)

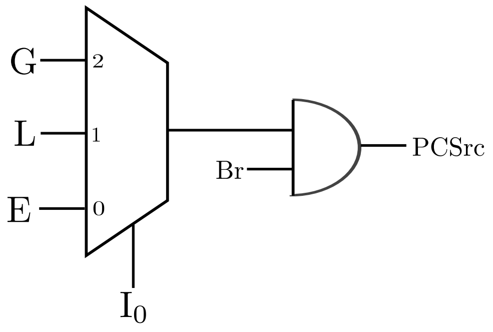


Figure 6.11: Proposed Design of Branching Unit (BU)

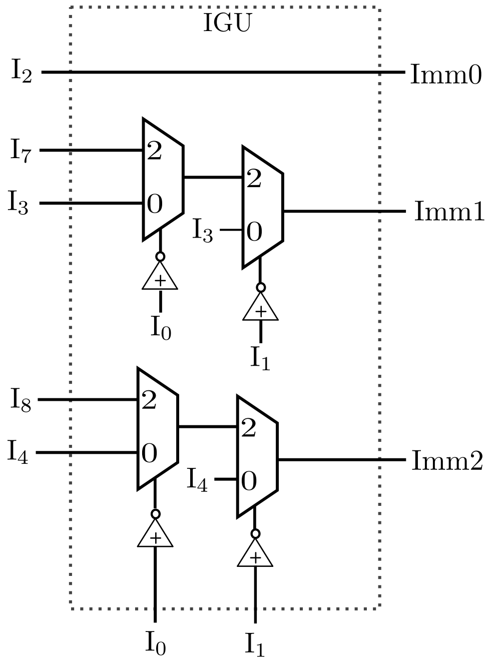


Figure 6.12: Proposed Design of Immediate Generate Unit (IGU)

zeroth trit is Imm_0 is same as the I_2 trit of the instruction code. The first Imm_1 and second Imm_2 trits are implemented using two $2:1$ multiplexers with I_0 and I_1 as select lines and I_3, I_7 and I_8, I_4 as inputs respectively, where I_0,I_1,I_2, I_3,I_4, I_7 and I_8 are digits of the instruction code.

6.4 Simulation Results

6.4.1 Simulation Environment

All the proposed circuits are simulated in HSPICE, a circuit simulation tool by Synopsys. A standard CNFET model [21, 22, 26] by Stanford is used for the simulation of the circuits. The CNFETs used are configured to have three CNTs and a pitch value of 20nm. All the individual blocks of the proposed CNFET-based Ternary logic Processor like the TIF, TRF, TALU, TDM, and TCU are simulated by giving random test input patterns and their functionality is verified. The proposed TLP built by integrating these proposed individual blocks is also simulated in HSPICE. Performance parameters like power consumption, propagation delay, PDP, and number of CNFETs required are calculated for the TLP and the individual blocks of the TLP. In addition, the functionality of the proposed 3-trit TLP is verified by building a Simulink model. The working of the TLP is verified by running a few benchmark programs.

6.4.2 Results and Discussion

6.4.2.1 Ternary Instruction Fetch Unit

The proposed Ternary Instruction Fetch unit was simulated using CNFETs in HSPICE and its performance parameters were calculated. The TIF unit design consists of the implementation of a PC register, an adder and a ternary Instruction Memory (TIM). The design of the TIF unit using CNFETs is as described in the Section 6.3.3. There are two inputs given to the TIF unit, 'Clk' and 'reset'. And the output of the TIF is a 9-trit instruction code. in order to test theTIF implementation, the following was

Table 6.3: Performance parameters for various blocks of TLP

Design	**Power (uW)**	**Delay (ps)**	**PDP ($\times10^{-17}$)**
Proposed TIF	25.05	312.00	781.56
Proposed TRF	4.51	10.70	4.82
Proposed TALU	53.07	82.90	439.95
Proposed TDM	1.70	12.50	21.25

done :

- First, 9 out of the 27 locations of the TIM were initialised with different instructions, and the rest of the locations were initialised to 'nop' instructions (addi t0,t0,0).
- The 'clock' and 'reset' were given as inputs to both the PC register and the TIM. The reset was used to initialise the TIM to the required values and the PC to zero.
- The PC value is incremented for every positive edge of clock, using the adder, enabling it to point to the next instruction.
- The output, that is, the 9 different 9$-$trit instructions were obtained at the TIM output one for every clock cycle.

The inputs were given for about 40ns and the average power consumed by the TIF unit and the propagation delay was calculated. As shown in Table6.3 the TIF consumed 25.05 uWatts of power. The delay was calculated form a particular rising edge of clock to the instruction code output. The delays were calculated for all 9-trit instruction digits with respect to clock for all the 9 instructions and the maximum delay was reported as 312.0 ps as in Table 6.3. Also, the total number of CNFETs required for building the TIF is found to be 6064.

Table 6.4: Comparision of Simulation results for existing and proposed TRF and TDM

Design	**Power (uW)**	**Delay (ps)**	**PDP(x10^{-17})**
TRF			
Existing TRF [80]	83.90	12.81	107.39
Proposed TRF	4.51	10.70	4.82
TDM			
Existing TDM [16]	3.40	17.10	114.92
Proposed TDM	1.70	12.50	21.25

6.4.2.2 Ternary Register File

The CNFET-based TRF is implemented in HSPICE using the ternary registers, write decoder and multiplexers as described in Section 6.3.4. The functionality of the TRF was verified as follows:

- The ternary registers (t0-t8) were initialized to zeros.
- The inputs that are the two source register numbers, one write register number and one write data were given to the TRF to obtain the data corresponding to the read registers.
- The data was written into the two registers by making the regwrite signal as 1 and then the data written was read out.

The average power consumed by the TRF for random inputs given was 4.51 uWatts as shown in Table 6.3. The delay (read delay) was calculated from the data read out to the register number input that was given for various cases and the worst-case delay was found to be 10.7 ps. A total of 626 CNFETs were required to build the proposed TRF.

Remark 7 : The proposed TRF that uses the ternary D-flipflop in [78] is compared to the existing TRF that uses the static D-latch in [80]. The comparison of performance is shown in Table 6.4 . It is observed that the proposed TRF shows improvement of upto 94.6 % and 55.16% in power and PDP, respectively.

6.4.2.3 Ternary Data Memory

The proposed TDM of size $27X3$ is simulated in HSPICE and its performance is measured. The TDM implementation consists of a decoder, two multiplexers and a memory build using ternary SRAM as shown in Figure 6.9. The simulation of TDM was carried out as follows:

- First, the words of the TDM were initialised with the required data or with zeros.
- The inputs to the TDM are the write data, the write data address and the control signals 'memrd' and 'memwr'.
- The data is written into or read from the words in the TDM using the control signal. The data read from the word is the output of the TDM.

The inputs were given for around 90ns and the average power was calculated to be 1.7 uWatts as shown in Table 6.3. Four read and four write operations were performed on the TDM and the functionality was verified by the correct reading out of the data written into the TDM. The read delay is calculated from the data read out to the input data address for different read operations and the worst-case delay was found to be 12.5 ps.

Remark 8 : The proposed TDM is also compared to the existing TDM in Table 6.4. The proposed TDM uses the buffer-based single cell SRAM proposed in [14], and the existing TDM is implemented using the buffer-based single-cell SRAM proposed in [16]. The proposed TDM shows an improvement of upto 50% in power and upto 74.70% in PDP when compared to the existing TDM. The main reason for the power improvement is the use of a power efficient ternary SRAM for the proposed TDM. The total number of CNFETs required to build the TDM is 3360.

6.4.2.4 Ternary Arithmetic and Logic Unit

The TALU is implemented in HSPICE and the performance parameters are calculated. The simulation is done as follows:

- The two 3-trit data values (operands), A and B are taken as inputs to the TALU.
- The select lines S_0S_1 are also given as inputs to select the desired function that the TALU needs to perform. And the output of the TALU is $T_0T_1T_2$.

The input is given for around 50 ns and all the select cases of the TALU are given and the output is verified. The average power consumed by the proposed 3-trit TALU is 53.07 uWatts and the delay is calculated for the different digits of the output T with respect to changes input A, B and select S. The worst-case delay is found to be 82.9 ps as shown in Table 6.3. A total of 4268 CNFETs are required to build the TALU.

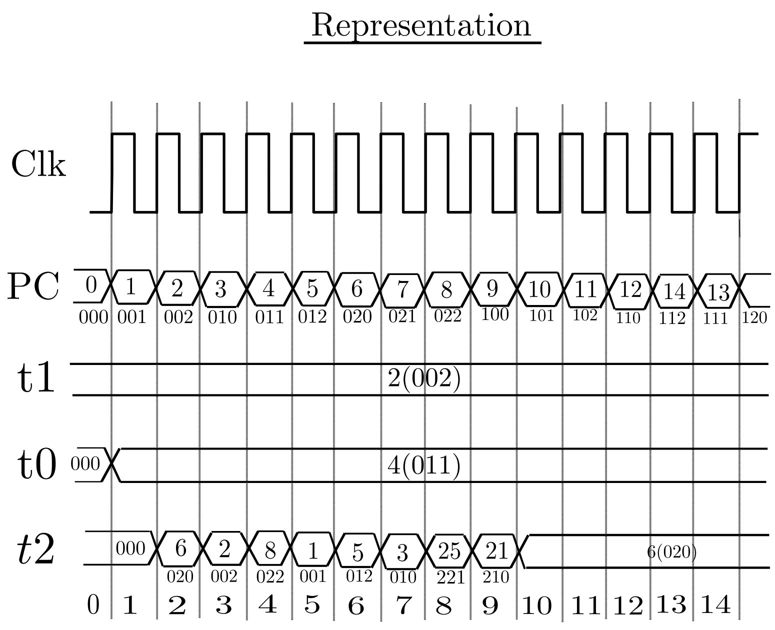
6.4.2.5 Ternary Logic Processor

The above blocks are used to implement the proposed TLP in HSPICE. The TIM of the TLP is loaded with a set of instructions(corresponding to a given program) that are defined in the ISA. The clock and reset are given as inputs to the TLP. Also, the TDM is initialized with some data as required by the program that is to be run. The output of the TLP is verfied by checking the correct writing of the desired results in the registers in the TRF and in the SRAM in TDM.

Initially, the ternary processor is simulated by running all 14 proposed instructions on it. The instructions proposed in Table 6.1 are loaded into the TIM and the TLP is simulated by giving a clock as input. The instruction codes loaded in the TIM for this are as shown in Program 1 of the Table 6.5 . The waveforms obtained by simulation of this TLP running the Program 1 is as shown in Figure 6.13. Here, 'Clk' represents the clock given to the TLP which has a clock period of 4 ns and is given for a total time period of around 60 ns. $PC_2PC_1PC_0$indicate the 3-trit PC register value that points to the current instruction to be executed in the TIM. The first two locations of the TDM are initialised with data values of 4(011) and 2(002) respectively, which are loaded into the registers t0 and t1 of the TRF, using the 'tlw' instruction. This constitutes the first 2 lines of the Program-1. The register t2 is taken as the destination resgister. And next all the R-type operations are performed on data in registers t0 and t1 and the result obtained is verified by checking the contents of the register t2. For example,

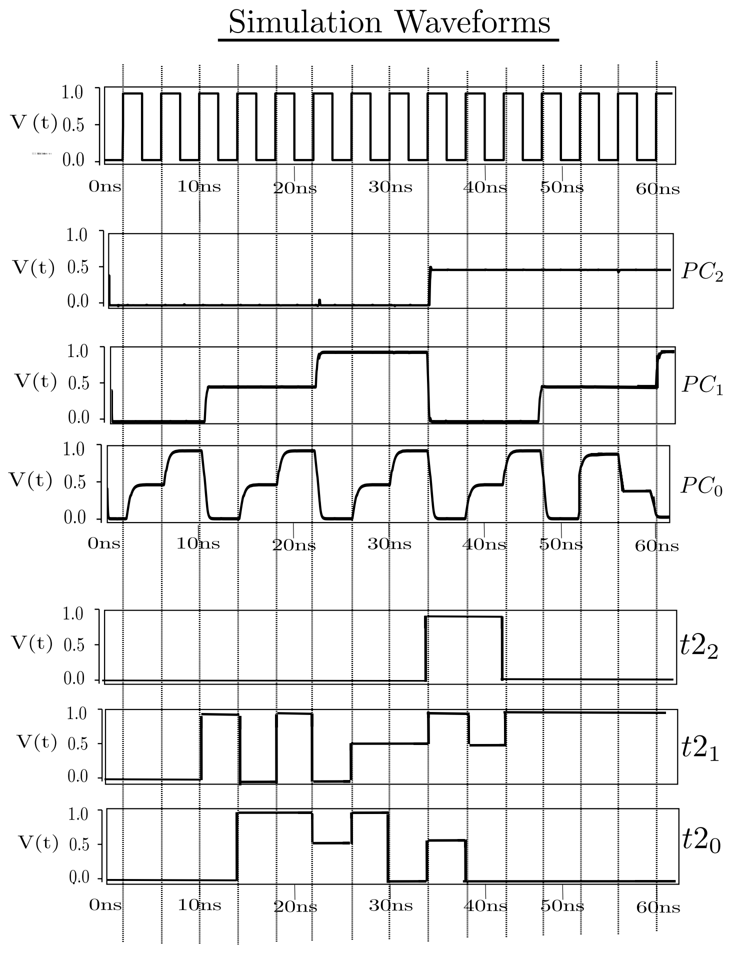
Table 6.5: Programs run on the Proposed TLP

Program 1- All Instructions		**Program 2- GCD**		**Program 3-Fibonacci**		**Program 4-Max code**	
Instruction	**Code**	**Instruction**	**Code**	**Instruction**	**Code**	**Instruction**	**Code**
lw t0,0(t0)	000000011	lw t1,0(t0)	000001011	lw t4,0(t0)	000011011	lw t1,0(t0)	000001011
lw t1,1(t1)	000101111	lw t2,1(t0)	000002111	L1:addi t1,t1,0	000101010	addi t2,t2,1	000202110
add t2,t0,t1	010002000	L1:beq t1,t2,L2	020102222	addi t2,t2,1	000202110	L1:bgt t2,t1,L2	010202120
sub t2,t0,t1	010002100	bgt t1,t2,020	020102020	addi t0,t0,1	000000110	lw t3,0(t2)	000210011
mul t2,t0,t1	010002200	sub t2,t2,t1	010202100	beq t4,t2,L2	021111222	bgt t4,t3,L1	101100120
and t2,t0,t1	010002001	beq t0,t0,L1	000000222	addi t3,t3,0	001010010	addi t4,t3, 0	001011010
or t2,t0,t1	010002101	sub t1,t1,t2	020101100	beq t3,t4,L3	111012022	beq t0,t0,L1	000000122
xor t2,t0,t1	010002201	beq t0,t0,L1	000000222	sw t1,0(t0)	010000012	L2:sw t4 0(t2)	110200012
nand t2,t0,t1	010002002	L2:sw t1,2(t0)	010000212	addi t0,t0,1	000000110		
nor t2,t0,t1	010002102			add t5,t1,t2	020112000		
addi t2,t0,2	000002210			addi t1,t2,0	000201010		
sw t2,0(t3)	021000012			addi t2,t5,0	001202010		
bgt t0,t1,112	010011220			addi t3,t3,1	001010110		
blt t1,t0,120	000112021			L2:beq t0,t0,L1	000002022		
beq t2,t2,111	020211122			L3:sw t1,0(t0)	010000012		



Instructions Index				
0-tlw t0,0(t0)	3-tsub t2,t0,t1	6-tor t2,t0,t1	9-tnor t2,t0,t1	12-tbgt t0,t1,112
1-tlw t1,1(t1)	4-tmul t2,t0,t1	7-txor t2,t0,t1	10-taddi t2,t0,2	13-tblt t2,t0,120
2-tadd t2,t0,t1	5-tand t2,t0,t1	8-tnand t2,t0,t1	11-tsw t2,0(t3)	14-tbeq t2,t2,111

(a) Representation



(b) Simulation Waveforms

Figure 6.13: Representation and the Simulation waveforms for the proposed TLP running Program-1 containing all instructions

Table 6.6: Simulation Results for Power consumed by the TLP to run benchmark programs

Program	Power consumed(in uW)
GCD of two numbers	79.97
Max of array of numbers	84.48
Fibonacci Series	85.15

the third instruction adds values in t0(4(011)) and t1(2(002)) and places the result in t2 (6(020)). The 3-trit representation of register t2 and the PC register are shown on the left hand side of the Figure 6.13 and their corresponding waveforms for each trit (digit) are shown on the right hand side.

The proposed TLP is also simulated to run three benchmark programs of calculating the GCD (Greatest Common Divisor) of two numbers, the Fibonacci series generation and calculating the maximum of an array of numbers. These programs are presented in the Table 6.5, as Program 2, Program 3 and Program 4 respectively. The instructions corresponding to there three programs are loaded in the TIM and the processor is run to obtain the desired results. The power consumed to run each of these programs on the TLP is calculated as shown in Table 6.6. Also, the Table6.5 shows the set of instructions, using the proposed ISA, corresponding to these three benchmark programs and their codes. These benchmark programs are specifically selected since they constitute of all of 14 proposed instructions and we were able to verify the proper functioning of the TLP for all the instructions.

In addition to this, the functionality of the proposed CNFET-based TLP is verified by building a Simulink Model of the processor. This functional verification is done for the all the instructions and the three benchmark programs of GCD (Greatest-Common-Divisor), max of two numbers and Fibonacci series.

The CNFET count that is number of transistors required to implement the individual blocks of the ternary processor like the ternary Instruction memory, register file , ALU, Data memory and the entire 3-trit ternary processor is presented in Table 6.7. Also, a binary processor of data size 5-digits(5-bits) is implemented in HSPICE for comparison with the proposed 3-digit(3-trit) processor and the two designs are com-

Table 6.7: CNFET count for the design of ternary processor

Block	**CNFETs count**
Ternary Instruction Fetch unit	$3,888$
Ternary Register File	708
Ternary ALU	1958
Ternay Data Memory	2160
Ternary Control unit	82
Other units (immediate unit, branch unit, muxes)	268

Table 6.8: Comparison of Ternary vs Binary processor designs

Design	**Proposed Ternary Processor**	**Equivalent Binary Processor**
CNFETs count	9064	16814
Power (in milli Watts)	0.079	7.17

pared in terms of power and number of transistors. The binary processor is designed using binary logic inverters, logic gates, and other modules like adder, subtractor, etc. All circuits are designed using CNTFET of single chirality of $(19, 0)$ to implement binary logic. The adder is a 5-bit ripple carry adder. For the multiplier, we have designed a $4X4$ array multiplier. The comparator is a 5-digit comparator that compares two 5-digit values and gives a three-bit output of l(lesser) e(equal) and g(greater) respectively. A simple code for the calculation of GCD of two numbers is run on both the ternary and binary processor and the designs are compared as presented in Table 6.8 and the ternary processor is therefore proved to be better than its binary counterpart.

6.5 Conclusions

This chapter presents the design and implementation of a 3-trit Ternary Logic Processor using CNFETs. The proposed single-cycle processor operates on 3-trit ternary data and can execute 14 different instructions of 9-trit length. The paper proposes an Instruction Set Architecture and presents the Micro-Architecture of the processor, including the design of the various blocks such as the Ternary Instruction Fetch unit, Ternary Register File, Ternary ALU, Ternary Data memory, and Ternary Control Unit, implemented using CNFETs. The performance parameters, such as power, delay, and the number of CNFETs, are calculated for each block of the processor. The proposed CNFET-based TLP design is validated by running a few programs on it and the functionality of the TLP is also verified using a Simulink model. The design of the CNFET-based Ternary processor proposed in this work provides a framework for designing more advanced Ternary processors that can handle larger data and include pipelining for better performance as a future scope of this work.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

This thesis presented the design and implememtation of a CNFET-based ternary logic processor. The proposed ternary processor is a single-cycle processor which operates on 3-digit ternary data. The circuit level design of each of the components of this processor is presented where every circuit is built using CNFETs. Initially, a ternary ALU is designed using a power efficient 2:1 multiplexer based approach. This ALU is implemented using a new architecture that eliminates the need of the decoder stage at the input making the design less complex in terms of number of transistors used when compared to the existing ternary ALU designs. Also, the 2:1 multiplexer based design approach for designing the functional modules of the ALU accounts for power improvement in the overall design.

Next, ternary sequential circuits like ternary D-flipflops are designed using four new design methodologies. One of these designs of a ternary D-flipflop that is built using a buffer-STI-based design approach consumes the least amount of power when compared to all other designs. This ternary D-flipflop design is used to implement a 3-digit ternary register which is further used to build the ternary register file that is used in the ternary processor design. Memory which is crutial to processor design is also implemented in this work. Three new power-efficient designs for a ternary single-cell SRAM are proposed. Two of these designs are cycle operator-based and one design is buffer-based. All the designs show considerable improvement in power as they use the Vdd/2-based design approach for building the ternary buffer and the cycle-operators. The proposed buffer-based single-cell SRAM design is used further to implement the ternary Instruction and data memories in the ternary processor design.

Finally, a power-efficient 3-digit ternary logic processor is designed using the proposed ternary ALU designed in the begining, the ternary register file which used the registers that are inturn built using the proposed ternary D-flipflops and the ternary instruction memory and data memory using the proposed low-power buffer-based ternary SRAM design. An Instruction set architecture is designed for the processor and the micro-architecture is proposed. The blocks other than those mentioned above which are required for the processor are the ternary control unit, branching unit and immediate generation unit. The design of these blocks is also presented. The final design of the 3-digit processor that can run 14 different instructions is implemented in HSPICE and its functionality is verified by running a few standard programs on it. The overall power consumed by the ternary processor for each of these programs is measured and noted. In addition to this the ternary processor design is also checked for its functional correctness by designing a Simulink model for the processor. In conclusion, this thesis demonstrated the design of a ternary logic processor and its individual blocks built entirely using CNFETs.

7.2 Future Scope

As the implementation of ternary logic circuits necessitates multiple thresholds, alternative device technologies to CNFET that can implement multiple thresholds, may be investigated in the future scope of this study. This work shows simulation of various ternary logic circuits. Further to this, synthesis techniques for these ternay circuits can be proposed.

Memory substantially contributes to the overall performance and latency of the processor. Therefore, designs for ternary memory that are more efficient than those presented here can be explored and implemented.

The ternary logic processor designed in this work can handle $3-$trit data. As an extension of this research, the processor can be designed to handle larger $n-$trit data. For this there is need to extend the proposed TALU to $n-$trit TALU and also extend the size of the register file and the memory of the processor. There is also a possibilty of proposing a more efficient MIPS, RISC type of instruction set architecture for the ternary logic processor.

Also the ternary processor designed in this work is a single -cycle processor which can be further utilised to build an advanced pipelined ternary processor.

Bibliography

[1] M. Horowitz, “Scaling, power and the future of cmos,” in *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID’07)*, pp. 23–23, 2007.

[2] E. Mollick, “Establishing moore’s law,” *IEEE Annals of the History of Computing*, vol. 28, no. 3, pp. 62–75, 2006.

[3] G. Hills, M. G. Bardon, G. Doornbos, D. Yakimets, P. Schuddinck, R. Baert, D. Jang, L. Mattii, S. M. Y. Sherazi, D. Rodopoulos, R. Ritzenthaler, C. S. Lee, A. V. Y. Thean, I. Radu, A. Spessot, P. Debacker, F. Catthoor, P. Raghavan, M. M. Shulaker, H. S. Wong, and S. Mitra, “Understanding Energy Efficiency Benefits of Carbon Nanotube Field-Effect Transistors for Digital VLSI,” *IEEE Transactions on Nanotechnology*, vol. 17, no. 6, pp. 1259–1269, 2018.

[4] G. Hills, C. Lau, A. Wright, S. Fuller, M. D. Bishop, T. Srimani, P. Kanhaiya, R. Ho, A. Amer, Y. Stein, D. Murphy, Arvind, A. Chandrakasan, and M. M. Shulaker, “Modern microprocessor built from complementary carbon nanotube transistors,” *Nature*, vol. 572, no. 7771, pp. 595–602, 2019.

[5] S. Lin, Y. B. Kim, and F. Lombardi, “CNTFET-based design of ternary logic gates and arithmetic circuits,” *IEEE Transactions on Nanotechnology*, vol. 10, no. 2, pp. 217–225, 2011.

[6] J. Liang, L. Chen, J. Han, and F. Lombardi, “Design and evaluation of multiple valued logic gates using pseudo N-Type carbon nanotube FETs,” *IEEE Transactions on Nanotechnology*, vol. 13, no. 4, pp. 695–708, 2014.

[7] R. F. Mirzaee, M. H. Moaiyeri, M. Maleknejad, K. Navi, and O. Hashemipour, "Dramatically low-transistor-count high-speed ternary adders," *Proceedings of The International Symposium on Multiple-Valued Logic*, pp. 170–175, 2013.

[8] R. F. Mirzaee, K. Navi, N. Bagherzadeh, R. Faghih Mirzaee, K. Navi, and N. Bagherzadeh, "High-Efficient Circuits for Ternary Addition," *VLSI Design*, vol. 2014, 2014.

[9] R. F. Mirzaee and N. Farahani, "Design of a Ternary Edge-Sensitive D FFF for Multiple-Valued Sequential Logic," *Journal of Low Power Electronics*, vol. 13, no. 1, pp. 36–46, 2017.

[10] S. L. Murotiya and A. Gupta, "Design of High Speed Ternary Full Adder and Three-Input XOR Circuits Using CNTFETs," *2015 28th International Conference on VLSI Design*, pp. 292–297, 2015.

[11] S. L. Murotiya and A. Gupta, "Hardware-efficient low-power 2-bit ternary ALU design in CNTFET technology," *International Journal of Electronics*, vol. 103, no. 5, pp. 913–927, 2016.

[12] S. Gadgil and C. Vudadha, "Design of cntfet-based ternary alu using 2:1 multiplexer based approach," *IEEE Transactions on Nanotechnology*, vol. 19, pp. 661–671, 2020.

[13] S. Karthikeyan, M. C. Karan Reddy, and P. R. Monica, "Design of CNTFET-Based ternary control unit and memory for a ternary processor," *2017 International Conference on Microelectronic Devices, Circuits and Systems, ICMDCS 2017*, vol. 2017-Janua, no. 1, pp. 1–4, 2017.

[14] S. Gadgil, G. N. Sandesh, and chetan Kumar V, "Power Efficient Designs of CNTFET-Based Ternary SRAM," 3 2023.

[15] Y. Shrivastava and T. K. Gupta, "Design of compact reliable energy efficient read disturb free 17t cnfet ternary s-ram cell," *IEEE Transactions on Device and Materials Reliability*, vol. 21, no. 4, pp. 508–517, 2021.

[16] B. Srinivasu and K. Sridharan, “Low-power and high-performance ternary sram designs with application to cntfet technology,” *IEEE Transactions on Nanotechnology*, vol. 20, pp. 562–566, 2021.

[17] S. Lin, Y.-B. Kim, and F. Lombardi, “Design of a ternary memory cell using cntfets,” *IEEE Transactions on Nanotechnology*, vol. 11, no. 5, pp. 1019–1025, 2012.

[18] D. Kam, J. G. Min, J. Yoon, S. Kim, S. Kang, and Y. Lee, “Design and evaluation frameworks for advanced risc-based ternary processor,” in *2022 Design, Automation Test in Europe Conference and Exhibition (DATE)*, pp. 1077–1082, 2022.

[19] S. Etezadi and S. Hosseini, “Novel ternary logic gates design in nanoelectronics,” *Advances in Electrical and Electronic Engineering*, vol. 17, 09 2019.

[20] J. Appenzeller, “Carbon Nanotubes for High-Performance Electronics - Progress and Prospect,” *Proceedings of the IEEE*, vol. 96, pp. 201–211, Feb 2008.

[21] J. Deng and H. S. P. Wong, “A Compact SPICE Model for Carbon-Nanotube Field-Effect Transistors Including Nonidealities and Its Application-Part I: Model of the Intrinsic Channel Region,” *IEEE Transactions on Electron Devices*, vol. 54, pp. 3186–3194, dec 2007.

[22] J. Deng and H. S. P. Wong, “A Compact SPICE Model for Carbon-Nanotube Field-Effect Transistors Including Nonidealities and Its Application-Part II: Full Device Model and Circuit Performance Benchmarking,” *IEEE Transactions on Electron Devices*, vol. 54, pp. 3195–3205, Dec 2007.

[23] S. Fregonese, H. Cazin d’Honincthun, J. Goguet, C. Maneux, T. Zimmer, J.-P. Bourgoin, P. Dollfus, and S. Galdin-Retailleau, “Computationally efficient physics-based compact cntfet model for circuit design,” *IEEE Transactions on Electron Devices*, vol. 55, no. 6, pp. 1317–1327, 2008.

[24] G. Gelao, R. Marani, R. Diana, and A. G. Perri, "A semiempirical spice model for n-type conventional cntfets," *IEEE Transactions on Nanotechnology*, vol. 10, no. 3, pp. 506–512, 2011.

[25] R. Marani, G. Gelao, and A. G. Perri, "Modelling of carbon nanotube field effect transistors oriented to spice software for a/d circuit design," *Microelectronics Journal*, vol. 44, no. 1, pp. 33–38, 2013. Special Issue of Microelectronics Journal on the IEEE International MOS-AK/GSA Workshop on Compact Modeling 2010.

[26] S. University, *Stanford University CNFET model Website. Stanford University, Stanford, CA [Online]*, 2008. http://nano.stanford.edu/model.php?id=23.

[27] S. C. Hu, *Ternary digital systems.* 1967.

[28] B. Srinivasu and K. Sridharan, "Carbon nanotube FET-based low-delay and low-power multi-digit adder designs," *IET Circuits, Devices & Systems*, vol. 11, no. 4, pp. 1–13, 2016.

[29] C. Vudadha, P. S. Phaneendra, and M. B. Srinivas, "An efficient design methodology for CNFET based ternary logic circuits," *Proceedings - 2016 IEEE International Symposium on Nanoelectronic and Information Systems, iNIS 2016*, no. 1, pp. 278–283, 2017.

[30] C. Vudadha, S. P. Parlapalli, and M. B. Srinivas, "Energy efficient design of CNFET-based multi-digit ternary adders," *Microelectronics Journal*, vol. 75, no. September 2017, pp. 75–86, 2018.

[31] C. K. Vudadha and M. Srinivas, "Design methodologies for ternary logic circuits," *Proceedings of The International Symposium on Multiple-Valued Logic*, vol. 2018-May, pp. 192–197, 2018.

[32] R. A. Jaber, J. M. Aljaam, B. N. Owaydat, S. A. Al-Maadeed, A. Kassem, and A. M. Haidar, "Ultra-low energy cnfet-based ternary combinational circuits designs," *IEEE Access*, vol. 9, pp. 115951–115961, 2021.

[33] P. Keshavarzian and R. Sarikhani, “A novel CNTFET-based ternary full adder,” *Circuits, Systems, and Signal Processing*, vol. 33, no. 3, pp. 665–679, 2014.

[34] S. A. Ebrahimi, P. Keshavarzian, S. Sorouri, and M. Shahsavari, “Low Power CNTFET- Based Ternary Full Adder Cell for Nanoelectronics,” *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 2, pp. 291–295, 2012.

[35] K. Sridharan, S. Gurindagunta, and V. Pudi, “Efficient multiternary digit adder design in CNTFET technology,” *IEEE Transactions on Nanotechnology*, vol. 12, no. 3, pp. 283–287, 2013.

[36] S. K. Sahoo, K. Dhoot, and R. Sahoo, “High performance ternary multiplier using CNTFET,” in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, IEEE, jul 2018.

[37] B. Srinivasu and K. Sridharan, “Low-Complexity Multiternary Digit Multiplier Design in CNTFET Technology,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 8, pp. 753–757, 2016.

[38] S. Tabrizchi, A. Panahi, F. Sharifi, K. Navi, and N. Bagherzadeh, “Method for designing ternary adder cells based on cnfets,” *IET Circuits, Devices & Systems*, vol. 11, no. 5, pp. 465–470, 2017.

[39] S. Heo, S. Kim, K. Kim, H. Lee, S.-Y. Kim, Y. J. Kim, S. M. Kim, H.-I. Lee, S. Lee, K. R. Kim, S. Kang, and B. H. Lee, “Ternary full adder using multi-threshold voltage graphene barristors,” *IEEE Electron Device Letters*, vol. 39, no. 12, pp. 1948–1951, 2018.

[40] S. Musala, A. Valluri, P. Gurubrahmam, G. Meghana, and N. Yashoda, “Design of cntfet based ternary subtractor using unary operators,” in *2022 IEEE International Symposium on Smart Electronic Systems (iSES)*, pp. 378–383, 2022.

[41] F. Zahoor, F. A. Hussin, F. A. Khanday, M. R. Ahmad, I. M. Nawi, and S. Gupta, “Carbon nanotube field effect transistor and resistive random access memory based

2-bit ternary comparator," in *2020 8th International Conference on Intelligent and Advanced Systems (ICIAS)*, pp. 1–6, 2021.

[42] M. Takbiri, K. Navi, and R. F. Mirzaee, "Systematic transistor sizing of a cnfet-based ternary inverter for high performance and noise margin enlargement," *IEEE Access*, vol. 10, pp. 10553–10565, 2022.

[43] C. Vudadha and M. B. Srinivas, "Design of high-speed and power-efficient ternary prefix adders using cnfets," *IEEE Transactions on Nanotechnology*, vol. 17, no. 4, pp. 772–782, 2018.

[44] N. S. Soliman, M. E. Fouda, L. A. Said, A. H. Madian, and A. G. Radwan, "N-digits ternary carry lookahead adder design," in *2019 31st International Conference on Microelectronics (ICM)*, pp. 142–145, 2019.

[45] S. Kim, Y. Kang, S. Baek, Y. Choi, and S. Kang, "Low-power ternary multiplication using approximate computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 8, pp. 2947–2951, 2021.

[46] J. Yoon, S. Baek, S. Kim, and S. Kang, "Optimizing ternary multiplier design with fast ternary adder," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 2, pp. 766–770, 2023.

[47] A. Dhande and V. Ingole, "Design And Implementation Of 2 Bit Ternary ALU Slice," *Proc. Int. Conf. IEEE-Sci. Electron., Technol. Inf. Telecommun*, pp. 1–11, 2005.

[48] S. L. Murotiya and A. Gupta, "Design of CNTFET-based 2-bit ternary ALU for nanoelectronics," *International Journal of Electronics*, vol. 101, no. 9, pp. 1244–1257, 2014.

[49] K. L. Sharma Trapti, "Energy-efficient ternary arithmetic logic unit design in cntfet technology," *Circuits, Systems, and Signal Processing*, vol. 39, 2019.

[50] T. A. Irving, S. G. Shiva, and H. T. Nagle, "Flip-flops for multiple-valued logic," *IEEE Transactions on Computers*, vol. C-25, no. 3, pp. 237–246, 1976.

[51] N. Z. Haomin, Wu, "Research into ternary edge-triggered jkl flip-flop," *Journal of Electronics (China)*, 1991.

[52] Q. Y. Pengjun, Wang and Z. Xuesong, "Design of ternary low-power domino jkl flipâflop and its application," *Journal of Semiconductors*, vol. 33, p. 115008, 2012.

[53] Y. Kang, P. Wang, Y. Zhang, and G. Li, "Design of ternary pulsed reversible counter based on cnfet," in *2017 IEEE 12th International Conference on ASIC (ASICON)*, pp. 375–378, 2017.

[54] S. Kim, S.-Y. Lee, S. Park, and S. Kang, "Design of quad-edge-triggered sequential logic circuits for ternary logic," in *2019 IEEE 49th International Symposium on Multiple-Valued Logic (ISMVL)*, pp. 37–42, 2019.

[55] T. Uemura and T. Baba, "A three-valued d-flip-flop and shift register using multiple-junction surface tunnel transistors," *IEEE Transactions on Electron Devices*, vol. 49, no. 8, pp. 1336–1340, 2002.

[56] M. H. Moaiyeri, M. K. Q. Jooq, A. Al-Shidaifat, and H. Song, "Breaking the limits in ternary logic: An ultra-efficient auto-backup/restore nonvolatile ternary flip-flop using negative capacitance cntfet technology," *IEEE Access*, vol. 9, pp. 132641–132651, 2021.

[57] K. Rahbari and S. A. Hosseini, "Novel ternary d-flip-flap-flop and counter based on successor and predecessor in nanotechnology," *AEU - International Journal of Electronics and Communications*, vol. 109, pp. 107–120, sep 2019.

[58] M. H. Moaiyeri, M. Nasiri, and N. Khastoo, "An efficient ternary serial adder based on carbon nanotube fets," *Engineering Science and Technology, an International Journal*, vol. 19, no. 1, pp. 271–278, 2016.

[59] T. Sharma and L. Kumre, “Energy-Efficient Ternary Arithmetic Logic Unit Design in CNTFET Technology,” *Circuits, Systems, and Signal Processing*, 2019.

[60] T. Sharma and D. Sharma, “Energy efficient circuit design of single edge triggered ternary shift registers using cnt technology,” *IEEE Transactions on Nanotechnology*, vol. 22, pp. 102–111, 2023.

[61] P. Balla and A. Antoniou, “Low power dissipation mos ternary logic family,” *IEEE Journal of Solid-State Circuits*, vol. 19, no. 5, pp. 739–749, 1984.

[62] A. Heung and H. Mouftah, “Depletion/enhancement cmos for a lower power family of three-valued logic circuits,” *IEEE Journal of Solid-State Circuits*, vol. 20, no. 2, pp. 609–616, 1985.

[63] 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*, pp. 435–438, 2009.

[64] G. Cho and F. Lombardi, “Design and process variation analysis of cntfet-based ternary memory cells,” *Integration*, vol. 54, pp. 97–108, 2016.

[65] Y. Shrivastava and T. K. Gupta, “Design of high-speed low variation static noise margin ternary s-ram cells,” *IEEE Transactions on Device and Materials Reliability*, vol. 21, no. 1, pp. 102–110, 2021.

[66] Y. Choi, S. Kim, K. Lee, and S. Kang, “Design and analysis of a low-power ternary sram,” in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, 2021.

[67] S. Tabrizchi, S. Angizi, and A. Roohi, “Design and evaluation of a robust power-efficient ternary sram cell,” in *2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1–4, 2022.

[68] H. G. P. N. e. a. Shulaker, M., “Carbon nanotube computer,” *Nature*, pp. 1476–4687, 2013.

[69] C. K. Vudadha and M. Srinivas, “Design methodologies for ternary logic circuits,” in *2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL)*, pp. 192–197, 2018.

[70] H. Sirugudi, S. Gadgil, and C. Vudadha, “A novel low power ternary multiplier design using cnfets,” in *2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*, pp. 25–30, 2020.

[71] V. Sridevi and T.Jayanthy, “Minimization of cntfet ternary combinational circuits using negation of literals technique,” *Arabian Journal for Science and Engineering*, 2014.

[72] S. K. Sahoo, K. Dhoot, and R. Sahoo, “High performance ternary multiplier using CNTFET,” *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, vol. 2018-July, pp. 269–274, 2018.

[73] T. Sharma and L. Kumre, “Design of unbalanced ternary counters using shifting literals based d-flip-flops in carbon nanotube technology,” *Computers and Electrical Engineering*, vol. 93, p. 107249, 2021.

[74] S. Gadgil and C. Vudadha, “Design of cnfet-based low-power ternary sequential logic circuits,” in *2021 IEEE 21st International Conference on Nanotechnology (NANO)*, pp. 169–172, 2021.

[75] S. Kim, S.-Y. Lee, S. Park, K. R. Kim, and S. Kang, “A logic synthesis methodology for low-power ternary logic circuits,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 9, pp. 3138–3151, 2020.

[76] S. Lin, Y. Kim, and F. Lombardi, “Cntfet-based design of ternary logic gates and arithmetic circuits,” *IEEE Transactions on Nanotechnology*, vol. 10, no. 2, pp. 217–225, 2011.

[77] R. A. Jaber, A. Kassem, A. M. El-Hajj, L. A. El-Nimri, and A. M. Haidar, "High-performance and energy-efficient cnfet-based designs for ternary logic circuits," *IEEE Access*, vol. 7, pp. 93871–93886, 2019.

[78] S. Gadgil and C. Vudadha, "Novel design methodologies for cnfet-based ternary sequential logic circuits," *IEEE Transactions on Nanotechnology*, vol. 21, pp. 289–298, 2022.

[79] V. Prasad, A. Banerjee, and D. Das, "Design of ternary encoder and decoder using cntfet," *International Journal of Electronics*, vol. 109, no. 1, pp. 135–151, 2022.

[80] A. Mohammaden, M. E. Fouda, I. Alouani, L. A. Said, and A. G. Radwan, "Cntfet design of a multiple-port ternary register file," *Microelectronics Journal*, vol. 113, p. 105076, 2021.

Related Journal Publications

1. **S. Gadgil** and C. Vudadha, "Design of CNTFET-Based Ternary ALU Using 2:1 Multiplexer Based Approach", in *IEEE Transactions on Nanotechnology*, vol. 19, pp. 661-671, 2020, doi: 10.1109/TNANO.2020.3018867.

2. **S. Gadgil** and C. Vudadha, "Novel Design Methodologies for CNFET-Based Ternary Sequential Logic Circuits", in *IEEE Transactions on Nanotechnology*, vol. 21, pp. 289-298, 2022, doi: 10.1109/TNANO.2022.3184759.

3. **S. Gadgil**, G.N. Sandesh and C. Vudadha, "Power efficient designs of CNTFET-based ternary SRAM", *Microelectronics Journal* (2023), doi: 10.1016/j.mejo.2023.105884.

4. **S. Gadgil**, G. N. Sandesh and C. Vudadha, "Design and Implementation of a CNTFET-Based Ternary Logic Processor", submitted to *Circuits, Systems and Signal Processing.* (In Review)

Related Conference Publications

1. P. Patel, N. Doddapaneni, **S. Gadgil** and C. Vudadha, "Design of Area Optimised, Energy Efficient Quaternary Circuits Using CNTFETs," 2019 *IEEE International Symposium on Smart Electronic Systems (iSES)* (Formerly iNiS), Rourkela, India, 2019, pp. 280-283, doi: 10.1109/iSES47678.2019.00069.

2. H. Sirugudi, **S. Gadgil** and C. Vudadha, "A Novel Low Power Ternary Multiplier Design using CNFETs," 2020 *33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*, Bangalore, India, 2020, pp. 25-30, doi: 10.1109/VLSID49098.2020.00022.

3. **S. Gadgil** and C. Vudadha, "Design of CNFET-based Low-Power Ternary Sequential Logic circuits," 2021 *IEEE 21st International Conference on Nanotechnology (NANO)*, Montreal, QC, Canada, 2021, pp. 169-172, doi: 10.1109/NANO51122.2021.9514328.

4. S. T, **S. Gadgil** and C. Vudadha, "Design of CNTFET-based Ternary Logic circuits using Low power Encoder," 2022 *IEEE International Symposium on Smart Electronic Systems (iSES)*, Warangal, India, 2022, pp. 142-147, doi: 10.1109/iSES54909.2022.00038.

Biographies

Candidate Biography

Sharvani Gadgil received her M.Tech (Master of Technology) degree in Embedded Systems from Jawaharlal Nehru Technological University, Hyderabad, India. She is currently pursuing her PhD degree at Birla Institute of Technology and Science (BITS)-Pilani, Hyderabad Campus, India. Her current research interests are CNFET-based MultiValued Logic Design and arithmetic circuits design.

Supervisor Biography

Chetan Vudadha received his Ph.D from Birla Institute of Technology and Science (BITS)-Pilani, India. He is currently working as Assistant Professor at BITS-Pilani, Hyderabad Campus. His research interests include CNFET-based Multi-Valued Logic Design, reversible logic and quantum computation and arithmetic circuits design.