Formulation and Design of Useful Logic Gates Using Quaternary Algebra

Sarah Nahar Chowdhury, Asif Faiyaz, Khandakar Mohammad Ishtiak

Abstract— Multivalued logic is preferable over conventional binary logic operations for speed optimization and information handling capacity. In this paper, a set of contemporary special operators using quaternary logic have been derived from basic gates. These novel special logic gates along with their corresponding equations and truth table can be implemented to function individually and in logic blocks resulting in a reduction of circuit complexity and better speed processing in integrated circuit technology.

Index Terms— BITSWAP, INWARD, multivalued logic, OUTWARD, quaternary algebra, qudit, Sum of products.

1 Introduction

Binary logic can be professed as the pivot of controlling the embedded fundamental and digital system of modern times. Due to its easy accessibility and widespread use of logic circuits, it is extremely quintessential in terms of its applications from simple circuits to complex one. Operating with binary logic seems controlling the real world with computers and that an alternative improved approach with a better usage of transmission path, storage and processing of large amount of information in digital signal processing seems impossible. Yet, the inclusion of a 'don't care' operator describes the importance of an intermediate or neutral stage even in binary logic. In addition to the existing binary values {0,1} we need a three valued {0,1,X} operands for constructing the Karnaugh map or applying the tabular method [1],[2]. Multivalued logic system introduces new operators in addition to binary values {0,1} as a solution to solving these intermediate stages. It is a proposed extension of the idea that n valued logic can be used instead of two logical values (i.e. true or false, logic high or low) where n>2.

Perhaps one of the most tangible immediate benefits of higher-radix approaches like quaternary logic lie in their potential for reduction of the wiring congestion both on-chip and between chips. Using a single conductor to transmit three or more discrete voltages or current values allows for greater information content per wire and thus results in a circuit with reduced conductors and logic gates than the binary-valued counterpart**.** By providing an increase in the information content per wire we can subsequently decrease the interconnection cost, complexity and area also. Furthermore, multivalued number systems, such as ternary and quaternary systems with a radix more than '2' (p>2) are obviously expected to have larger information handling and storage capacities.

- *Sarah Nahar Chowdhury, Ahsanullah University of Science and Technology (AUST), Bangladesh. E-mail: sarahnahar55@gmail.com*
- *Asif Faiyaz, Ahsanullah University of Science and Technology (AUST), Bangladesh. E-mail: asif.faiyaz@gmail.com*
- *Khandakar Mohammad Ishtiak, Ahsanullah University of Science and Technology (AUST), Bangladesh. E-mail: ishtiak.eee@aust.edu*

Many complex logic systems and associated algebras are possible due to the exponentially increasing number of operators with respect to the cardinality of the quaternary logic values. Arithmetic logic circuits mainly based upon addition circuitry can be constructed where operations on each operand can be performed simultaneously using multiple values and thus overcoming the effect of overall circuit delay which is subsequently a function of the length of inherent carry digit propagation characteristic chain.

Quaternary algebra can be used in response to the necessity of representation of intermediate values in Hardware Description Languages (HDL) and Symbolic Trajectory Evaluation (STE) algorithm which is used for verification of errors before the circuit is designed in EDA-CAD [1]. Quaternary algebra can even be applied to optical computing for novel photonics structures through optical quaternary logic.

Although quaternary algebra is not as widespread as binary and does not have all the logic gates, earlier works suggest some proposed logic gates for quaternary. In this paper, we demonstrate some new logic gates contrary to the earlier proposed logic operators.

2 Quaternary Algebra

Quaternary algebra is a subsidiary of multi valued logic. While multi valued logic deals with infinite number of values as discrete variables quaternary algebra is based upon four discrete variables including the binary values. Quaternary states {0,1,2,3} with a set of operators and axioms are used to define quaternary algebra. Each of the quaternary states {0,1,2,3} has its two bits binary equivalents 00 (absolute low), 01 (intermediate low), 10 (intermediate high) and 11 (absolute high). Each of the quaternary bits is called 'qudit'[1],[3], when expressed in numbers and can also be indicated by two binary digits 'a1' and 'a0', respectively.

They are inscribed and packed together using the following notion A={a1,a0} and the term '2a1 + a0' denotes the magnitude of the variable 'A' in decimal system [4]. Quaternary states are sub categorized into symmetrical and asymmetrical

based upon their position of bits. If the bits of the binary equivalent of quaternary states interchange their position and still the quaternary state remain unchanged then they are known as symmetrical. Absolute states (0,3) are symmetrical as the change of bits in binary equivalent does not change the corresponding quaternary value. If the alternation of position of bits changes the corresponding binary value then they are known as asymmetrical. Intermediate states (1,2) are asymmetrical as interchanging the binary equivalents changes the quaternary state 1 to 2 and vice versa [5].

Quaternary algebra does not replace binary logic rather it broadens the applications of binary through high storage capacity and multitasking. That is why the logic gates derived in binary algebra are also applicable in quaternary algebra [4]. They are OR, AND, BUFFER, BASIC INVERTER, XOR, BASIC NAND, BASIC NOR and BASIC XNOR.

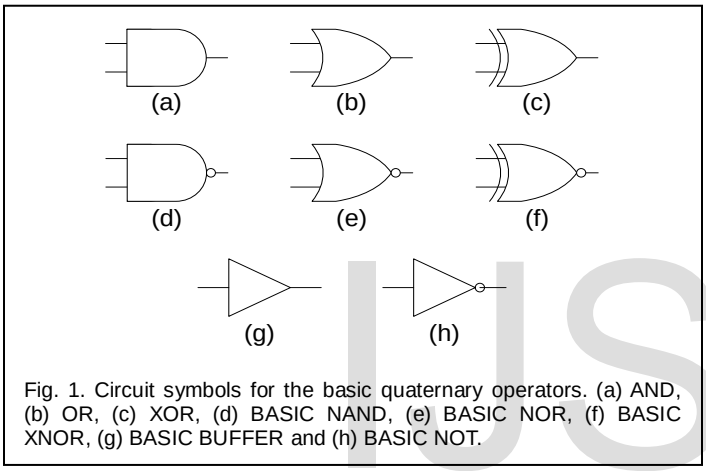

Fig. 1. Circuit symbols for the basic quaternary operators. (a) AND, (b) OR, (c) XOR, (d) BASIC NAND, (e) BASIC NOR, (f) BASIC XNOR, (g) BASIC BUFFER and (h) BASIC NOT.

TABLE 1
Basic Quaternary Multi-Input Operators Truth Table

Input		Output					
A	B	AND	OR	XOR	NAND	NOR	XNOR
0	0	0	0	0	3	3	3
0	1	0	1	1	3	2	2
0	2	0	2	2	3	1	1
0	3	0	3	3	3	0	0
1	0	0	1	1	3	2	2
1	1	1	1	0	2	2	3
1	2	0	3	3	3	0	0
1	3	1	3	2	2	0	1
2	0	0	2	2	3	1	1
2	1	0	3	3	3	0	0
2	2	2	2	0	1	1	3
2	3	2	3	1	1	0	2
3	0	0	3	3	3	0	0
3	1	1	3	2	2	0	1
3	2	2	3	1	1	0	2
3	3	3	3	0	0	0	3

In addition to the mainstream basic operators, a number of special logic gates like INWARD INVERTER, INWARD NAND, INWARD NOR, INWARD XNOR, OUTWARD INVERTER, OUTWARD NAND, OUTWARD NOR, OUTWARD XNOR, BINARY BITSWAP, BITSWAP AND, BITSWAP OR and BITSWAP XOR have been introduced in [3],[4] solving much more complex functions.

The mathematical definitions of special operators [5],[6] are given below:

INWARD INVERTER, $$a' = \begin{cases} \bar{a}.2\,; & a < 2 \\ \bar{a}+1; & a > 1 \end{cases} \quad (1)$$

OUTWARD INVERTER, $$\hat{a} = \begin{cases} \bar{a}+3; & a < 2 \\ \bar{a}.0\,; & a > 1 \end{cases} \quad (2)$$

BINARY BITSWAP, $$\tilde{a} = \begin{cases} \bar{a}; & a \ \ asymmetric \\ a; & a \ \ symmetric \end{cases} \quad (3)$$

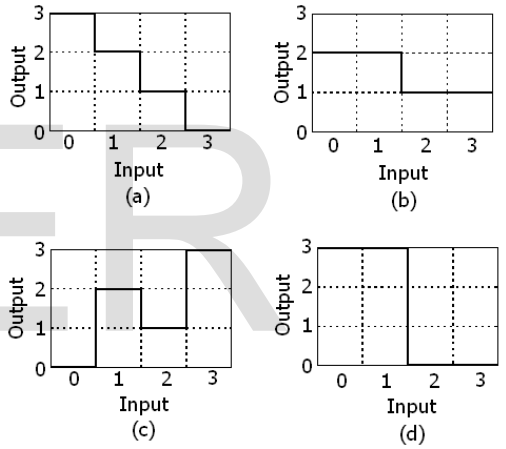

Fig. 2. Transfer functions of the quaternary inverters and inverter-like operators: (a) Basic Inverter; (b) Inward or Half Inverter; (c) Binary Bitswap; (d) Outward or Full Inverter.

Like binary logic, any function even for special logic operators can be implemented in quaternary algebra using the basic SOP II form that has been proposed in [4]. Although we have multiple min terms in quaternary algebra all these min terms are primarily converted into binary equivalents and then implemented in the following equations.

$$F1 \equiv a, f1 \quad (4)$$
$$F0 \equiv b, f\,0 \quad (5)$$

where, a and b are arbitrary binary values being converted into their quaternary equivalents and then both the functions are combined to form the required SOP format to represent all the functions for logic gates [4].

$$F = \sim F1.2 + F0.1 \quad (6)$$

The transformation of binary to quaternary equivalents can be done using the following transformation pairs in the truth table.

TABLE 2
BASIC TRUTH TABLE OF SOP II FORM

f0 →F	*f1 →F*
$x0 \equiv X \,.\, 1$	$x0 \equiv \sim X \,.\, 2$
$x0 \equiv X \,.\, 1$	$x0 \equiv \sim X \,.\, 2$
$x1 \equiv \sim X \,.\, 1$	$x1 \equiv X \,.\, 2$
$x1 \equiv \sim X \,.\, 1$	$x1 \equiv X \,.\, 2$

Quaternary algebra can be used as models for the initial design of logic circuits whether they are implemented with MVL signal levels or binary after being encoded. The most extolling part of quaternary algebra is that the interfacing of binary to quaternary and in reverse can be easily conducted using an encoder and this conversion of multi valued variables in quaternary algebra to binary is called encoding. Furthermore, the conversion can also be done directly when binary system has 2m inputs and 2n outputs. The 2m inputs are grouped as m quaternary inputs and then the 2n binary outputs are converted to n quaternary outputs using the transformation pair format used in SOP form. Moreover, all the logic blocks of quaternary are compatible with binary logic design making it a paragon for dual purpose.

3 PROPOSED IDEA

We know in case of ordinary binary input, NAND gate can be generated combining AND operator and NOT operator. NAND gate can be implemented in two ways. One is to fabricate AND gate first and NOT gate subsequently. Another way is to devise it by placing NOT gate first and then AND gate. Previously only one of the orders were followed in [3]- [6] for operators like INWARD INVERTER, INWARD NAND, INWARD NOR, INWARD XNOR, OUTWARD INVERTER, OUTWARD NAND, OUTWARD NOR, OUTWARD XNOR, BINARY BITSWAP, BITSWAP AND, BITSWAP OR and BITSWAP XOR in quaternary logic like Boolean algebra. But our proposed research and data showed that when we do INWARD and then NAND, the logic table obtained is not the same as the logic table obtained when we do NAND and then INWARD. The equations derived for each corresponding order also shows variance in comparison to each other. Only for BITSWAP operator, the complementary equations and logic tables of any order shows identical output. Earlier the same symbol and equation was used to demonstrate both the processes. But in our paper, we conclude our findings of different truth tables and equations for both the operations. The symbols used for each case has been mentioned below.

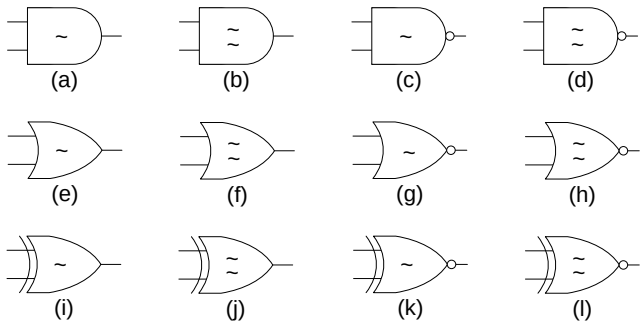

Fig. 3. Circuit symbols for the special quaternary operators and their derivatives: (a) BITSWAP AND, (b) AND BITSWAP, (c) BITSWAP NAND, (d) NAND BITSWAP, (e) BITSWAP OR, (f) OR BITSWAP, (g) BITSWAP NOR, (h) NOR BITSWAP, (i) BITSWAP XOR, (j) XOR BITSWAP, (k) BITSWAP XNOR, (f) XNOR BITSWAP.

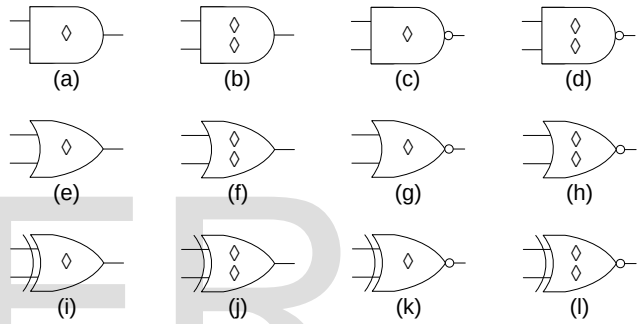

Fig. 4. Circuit symbols for the special quaternary operators and their derivatives: (a) INWARD AND, (b) AND INWARD, (c) INWARD NAND, (d) NAND INWARD, (e) INWARD OR, (f) OR INWARD, (g) INWARD NOR, (h) NOR INWARD, (i) INWARD XOR, (j) XOR INWARD, (k) INWARD XNOR, (f) XNOR INWARD.

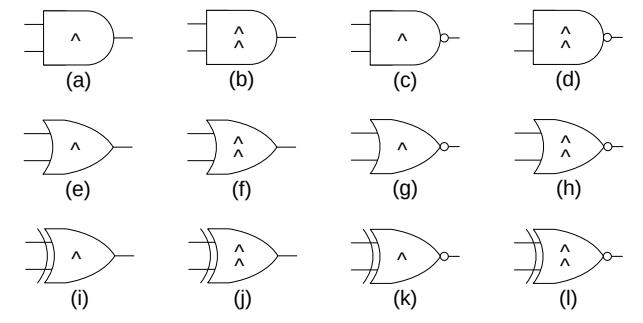

Fig. 5. Circuit symbols for the special quaternary operators and their derivatives: (a) OUTWARD AND, (b) AND OUTWARD, (c) OUTWARD NAND, (d) NAND OUTWARD, (e) OUTWARD OR, (f) OR OUTWARD, (g) OUTWARD NOR, (h) NOR OUTWARD, (i) OUTWARD XOR, (j) XOR OUTWARD, (k) OUTWARD XNOR, (l) XNOR OUTWARD.

Each logic gate can be represented by two symbols. The reason behind two discrete symbols used is to designate each operator having exclusive equations and logic table with a particular symbol. The respective equations derived are based on the SOP-II format that has been used in [4] and cannot by any means be interchanged with each other. Let's consider BITSWAP AND gate and the process of deriving the specific equation. The quaternary truth table of BITSWAP AND gate is given below:

TABLE 3
TABLE FOR BITSWAP AND GATE.

B \ A	0	1	3	2
0	0	0	0	0
1	0	2	2	0
3	0	2	3	1
2	0	0	1	1

Now, using the SOP-II format, we consider the cells which contains 1 or 3 as true for ba0 (BITSWAP AND). Similarly, we consider the cells which contain 2 or 3 as true for ba1. The remaining cells will contain 0. The K-map for ba0 is given in the following table:

TABLE 4
K – MAP FOR BITSWAPAND$_0$ FUNCTION.

$<b_1,b_0>$ \ $<a_1,a_0>$	0,0	0,1	1,1	1,0
0,0	0	0	0	0
0,1	0	0	0	0
1,1	0	0	**1**	**1**
1,0	0	0	**1**	**1**

From the K-map given above, we can write the following expressions:

$$ba_0\,(a_1,a_0,b_1,b_0) = a_1.b_1 \quad (7)$$

Now, we replace all binary variables from (7) and get (8):

$$BA_0\,(A,B) = (\sim A.\sim B).1 \quad (8)$$

Now, the K-map for ba1 is given below:

TABLE 5
K – MAP FOR BITSWAPAND$_1$ FUNCTION.

$<b_1,b_0>$ \ $<a_1,a_0>$	0,0	0,1	1,1	1,0
0,0	0	0	0	0
0,1	0	**1**	**1**	0
1,1	0	**1**	**1**	0
1,0	0	0	0	0

From the K-map we get the following expression:

$$ba_1 = a_0.b_0 \quad (9)$$

Then we replace all binary variables from (9) and get (10):

$$BA_1 = (\sim A.\sim B).2 \quad (10)$$

Now, we can combine (8) and (10) to get the complete function:

$$\begin{aligned} \text{BITSWAP AND (A,B)} &= (\sim A.\sim B).1 + (\sim A.\sim B).2 \\ &= (\sim A.\sim B).(1+2) \\ &= (\sim A.\sim B).3 \\ &= (\sim A.\sim B) \text{ OR } \sim (A.B) \end{aligned}$$

This is the final equation of BITSWAP AND gate. Similarly, we can use the above procedure to derive the following equations:

3.1 BITSWAP Operators

- BITSWAP AND, AND BITSWAP : $(\sim A.\sim B)$ OR $\sim (A.B)$
- BITSWAP NAND, NAND BITSWAP : $(\sim \overline{A} + \sim \overline{B})$ OR $\sim (\overline{A} + \overline{B})$
- BITSWAP OR, OR BITSWAP : $(\sim A + \sim B)$ OR $\sim (A + B)$
- BITSWAP NOR, NOR BITSWAP : $(\sim \overline{A}.\sim \overline{B})$ OR $\sim (\overline{A}.\overline{B})$
- BITSWAP XOR, XOR BITSWAP : $(\sim \overline{A}.\sim B + \sim A.\sim \overline{B})$
- BITSWAP XNOR, XNOR BITSWAP : $(\sim A.\sim B + \sim \overline{A}.\sim \overline{B})$

The truth table shows the comparison of output of special logic gates when two basic operators are fabricated in a particular order to form the special quaternary operators. From the relation we can conclude that the special operators do not follow the law of associatively and commutatively.

TABLE 6
TRUTH TABLE FOR SOME PROPOSED GATES CONTAINING BITSWAP INVERTER.

Input		Output					
A	B	Or Bitswap	Bitswap Or	Nor Bitswap	Bitswap Nor	And Bitswap	Bitswap And
0	0	0	0	3	3	0	0
0	1	2	2	1	1	0	0
0	2	1	1	2	2	0	0
0	3	3	3	0	0	0	0
1	0	2	2	1	1	0	0
1	1	2	2	1	1	2	2
1	2	3	3	0	0	0	0
1	3	3	3	0	0	2	2
2	0	1	1	2	2	0	0
2	1	3	3	0	0	0	0
2	2	1	1	2	2	1	1
2	3	3	3	0	0	1	1
3	0	3	3	0	0	0	0
3	1	3	3	0	0	2	2
3	2	3	3	0	0	1	1
3	3	3	3	0	0	3	3

TABLE 7
TRUTH TABLE FOR SOME PROPOSED GATES CONTAINING BITSWAP INVERTER.

Input		Output					
A	B	Nand Bitswap	Bitswap Nand	Xor Bitswap	Bitswap Xor	Xnor Bitswap	Bitswap Xnor
0	0	3	3	0	0	3	3
0	1	3	3	2	2	1	1
0	2	3	3	1	1	2	2
0	3	3	3	3	3	0	0
1	0	3	3	2	2	1	1
1	1	1	1	0	0	3	3
1	2	3	3	3	3	0	0
1	3	1	1	1	1	2	2
2	0	3	3	1	1	2	2
2	1	3	3	3	3	0	0
2	2	2	2	0	0	3	3
2	3	2	2	2	2	1	1
3	0	3	3	3	3	0	0
3	1	1	1	1	1	2	2
3	2	2	2	2	2	1	1
3	3	0	0	0	0	3	3

3.2 INWARD Operators

- INWARD AND :
$$(\sim A.\sim B).1+\left(\overline{A}.\overline{B}\right).2$$
- AND INWARD :
$$(\sim A.\sim B).1+\left(\overline{A}+\overline{B}\right).2$$
- INWARD NAND :
$$\left(\sim \overline{A}+\sim \overline{B}\right).1+(A+B).2$$
- NAND INWARD :
$$\left(\sim \overline{A}+\sim \overline{B}\right).1+(A.B).2$$
- INWARD OR :
$$(\sim A+\sim B).1+\left(\overline{A}+\overline{B}\right).2$$
- OR INWARD :
$$(\sim A+\sim B).1+\left(\overline{A}.\overline{B}\right).2$$
- INWARD NOR :
$$(\sim \overline{A}.\sim \overline{B}).1+\left(A.B\right).2$$
- NOR INWARD :
$$(\sim \overline{A}.\sim \overline{B}).1+\left(A+B\right).2$$
- INWARD XOR :
$$\left(\sim A.\sim \overline{B}+\sim \overline{A}.\sim B\right).1+(\overline{A}.B+A.\overline{B}).2$$
- XOR INWARD :
$$\left(\sim A.\sim \overline{B}+\sim \overline{A}.\sim B\right).1+(\overline{A}.\overline{B}+A.B).2$$
- INWARD XNOR :
$$\left(\sim \overline{A}.\sim \overline{B}+\sim .A.\sim B\right).1+(\overline{A}.\overline{B}+A.B).2$$
- XNOR INWARD :
$$\left(\sim \overline{A}.\sim \overline{B}+\sim .A.\sim B\right).1+(\overline{A}.B+A.\overline{B}).2$$

TABLE 8
TRUTH TABLE FOR SOME PROPOSED GATES CONTAINING INWARD INVERTER.

Input		Output					
A	B	Or Inward	Inward Or	Nor Inward	Inward Nor	And Inward	Inward And
0	0	2	2	1	1	2	2
0	1	2	2	1	1	2	2
0	2	1	3	2	0	2	0
0	3	1	3	2	0	2	0
1	0	2	2	1	1	2	2
1	1	2	2	1	1	2	2
1	2	1	3	2	0	2	0
1	3	1	3	2	0	2	0
2	0	1	3	2	0	2	0
2	1	1	3	2	0	2	0
2	2	1	1	2	2	1	1
2	3	1	1	2	2	1	1
3	0	1	3	2	0	2	0
3	1	1	3	2	0	2	0
3	2	1	1	2	2	1	1
3	3	1	1	2	2	1	1

TABLE 9
TRUTH TABLE FOR SOME PROPOSED GATES CONTAINING INWARD INVERTER.

Input		Output					
A	B	Nand Inward	Inward Nand	Xor Inward	Inward Xor	Xnor Inward	Inward Xnor
0	0	1	1	2	0	1	3
0	1	1	1	2	0	1	3
0	2	1	3	1	3	2	0
0	3	1	3	1	3	2	0
1	0	1	1	2	0	1	3
1	1	1	1	2	0	1	3
1	2	1	3	1	3	2	0
1	3	1	3	1	3	2	0
2	0	1	3	1	3	2	0
2	1	1	3	1	3	2	0
2	2	2	2	2	0	1	3
2	3	2	2	2	0	1	3
3	0	1	3	1	3	2	0
3	1	1	3	1	3	2	0
3	2	2	2	2	0	1	3
3	3	2	2	2	0	1	3

3.3 OUTWARD Operators

- OUTWARD AND :
$$(\sim \overline{A}.\sim \overline{B}).1+\left(\overline{A}.\overline{B}\right).2$$
- AND OUTWARD :
$$(\sim \overline{A}+\sim \overline{B}).1+\left(\overline{A}+\overline{B}\right).2$$
- OUTWARD NAND :
$$(\sim A+\sim B).1+\left(A+B\right).2$$
- NAND OUTWARD :
$$(\sim A.\sim B).1+\left(A.B\right).2$$
- OUTWARD OR :
$$(\sim \overline{A}+\sim \overline{B}).1+\left(\overline{A}+\overline{B}\right).2$$
- OR OUTWARD :
$$(\sim \overline{A}.\sim \overline{B}).1+\left(\overline{A}.\overline{B}\right).2$$
- OUTWARD NOR :
$$(\sim A.\sim B).1+\left(A.B\right).2$$
- NOR OUTWARD :
$$(\sim A+\sim B).1+\left(A+B\right).2$$
- OUTWARD XOR :
$$(\sim \overline{A}.\sim B+\sim A.\sim \overline{B}).1+\left(\overline{A}.B+A.\overline{B}\right).2$$
- XOR OUTWARD :
$$(\sim \overline{A}.\sim \overline{B}+\sim A.\sim B).1+\left(\overline{A}.\overline{B}+A.B\right).2$$
- OUTWARD XNOR :
$$(\sim \overline{A}.\sim \overline{B}+\sim A.\sim B).1+\left(\overline{A}.\overline{B}+A.B\right).2$$
- XNOR OUTWARD :
$$(\sim \overline{A}.\sim B+\sim A.\sim \overline{B}).1+\left(\overline{A}.B+A.\overline{B}\right).2$$

TABLE 10
TRUTH TABLE FOR SOME PROPOSED GATES CONTAINING OUTWARD INVERTER.

Input		Output					
A	B	Or Outward	Outward Or	Nor Outward	Outward Nor	And Outward	Outward And
0	0	3	3	0	0	3	3
0	1	3	3	0	0	3	3
0	2	0	3	3	0	3	0
0	3	0	3	3	0	3	0
1	0	3	3	0	0	3	3
1	1	3	3	0	0	3	3
1	2	0	3	3	0	3	0
1	3	0	3	3	0	3	0
2	0	0	3	3	0	3	0
2	1	0	3	3	0	3	0
2	2	0	0	3	3	0	0
2	3	0	0	3	3	0	0
3	0	0	3	3	0	3	0
3	1	0	3	3	0	3	0
3	2	0	0	3	3	0	0
3	3	0	0	3	3	0	0

TABLE 11
TRUTH TABLE FOR SOME PROPOSED GATES CONTAINING OUTWARD INVERTER.

Input		Output					
A	B	Nand Outward	Outward Nand	Xor Outward	Outward Xor	Xnor Outward	Outward Xnor
0	0	0	0	3	0	0	3
0	1	0	0	3	0	0	3
0	2	0	3	0	3	3	0
0	3	0	3	0	3	3	0
1	0	0	0	3	0	0	3
1	1	0	0	3	0	0	3
1	2	0	3	0	3	3	0
1	3	0	3	0	3	3	0
2	0	0	3	0	3	3	0
2	1	0	3	0	3	3	0
2	2	3	3	3	0	0	3
2	3	3	3	3	0	0	3
3	0	0	3	0	3	3	0
3	1	0	3	0	3	3	0
3	2	3	3	3	0	0	3
3	3	3	3	3	0	0	3

4 RESULT

In this paper we concluded that some quaternary logic gates which are formed of combination of two basic gates can be represented in multiple ways having individual salient features through equations and function tables. The result of the proposed paper shows that for some special gates like INWARD OR, INWARD NAND etc., the corresponding equations and function tables shows dissimilarity. On the other hand for proposed gates like BITSWAP, the output is same for any particular order. Our findings contain all the logic gates that has been derived so far with their proper notation, subsequent symbol, logic function and output table for their incomparability along with some similitude in case of few logic gates.

5 FUTURE WORKS

The logic gates improvised in this paper can be implemented for better designing purposes. We seek to implement these improved logic gates in adder, subtractor, max, min, comparator and elementary sequential circuits. The special logic gates derived in this paper can be further employed to express equations and functions in a complex form though effective minimization using SOP form. Furthermore, it is proposed that using these logic gates, quaternary algebra can be used in large scale logic devices with even coupled binary inputs for execution of addition purposes in the high performance microprocessor. With the advancement of VLSI technology and the development and invention of novel electron devices like Carbon Nanotube Transistor, FinFET, G4-FET, Silicon Nanowire FET, etc. it is therefore possible to integrate circuits with quaternary logic rather than binary for fast processing in the forthcoming future.

6 CONCLUSION

Although binary logic is utilized and prevalent in nearly all the logic circuits and multi valued logic circuits are rather complex in nature, yet quaternary algebra imposes some features which can provide immense benefit to the VLSI and quantum technology. Furthermore, quantum computing is an emerging topic and the application of quaternary logic will be able to simplify the quantum circuit design and analysis through designing an efficient quantum signal processing logarithm reduction in the number of individual quantum systems required to span the quantum memory. With the advancement of VLSI technology and the development and invention of novel electron devices like Carbon Nanotube Transistor, FinFET, G4-FET, Silicon Nanowire FET, etc. it is possible to integrate circuits with quaternary logic rather than binary for fast processing. We should not just limit our idea with Aristotle hypothesis of defining every event or happening with 'True' or 'False' or binary equivalent of {0, 1} but outreach to far more efficient concepts like quaternary algebra which will be able to conduit a greater advancement in modern technology.

REFERENCES

[1] Miller, D. Michael; Thornton, Mitchell A. (2008). *Multiple valued logic: concepts and representations*. Synthesis lectures on digital circuits and systems 12. Morgan & Claypool Publishers.ISBN 978-1-59829-190-2.

[2] Hurst, S. L; "Multiple-Valued Logic - Its Status and Its Future", Computers, IEEE Transactions onVol. C-33, Issue: 12, pp.1160-1179, Dec. 1984.

[3] Gaidhani, Y. A; Monica N. K.; "Design of Some Useful Logic Blocks Using Quaternary Algebra", CEE 2011, May 2011, India.

[4] I. Jahangir; A. Das; M. Hasan; "Formulation and Development of a Novel Quaternary Algebra", *Journal of Multiple-Valued Logic and Soft Computing*.

[5] Jahangir, I.; Hasan, D. M. N.; Reza, M. S.; "Design of some quaternary combinational logic blocks using a new logic system", TENCON 2009 - 2009 IEEE Region 10 Conference, pp. 1 - 6, Singapore, 23-26 Jan. 2009.

[6] Jahangir, I.; Hasan, D. M. N.; Reza, M. S.; "On the Design and Analysis of Quaternary Serial and Parallel Adders", IEEE TENCON 2010, November 2010, Japan.