

# MINIMIZATION OF TERNARY COMBINATIONAL CIRCUITS - A SURVEY

A.SATHISH KUMAR AND A.SWETHA PRIYA

Department of Electronics & Communication Engineering, Amrita Vishwa Vidyapeetham,  
Amrita School of Engineering, Bangalore, Karnataka – 560 035, India.

## Abstract:

This paper presents a novel method for defining, analyzing and implementing the basic combinational circuitry with minimum number of ternary multiplexers. Multiplexer is used as basic building gate to realize all the sequential and combinational circuitry which provides complete, concise, implementation-free description of the ternary functions involved. The method is useful in analyzing the complex ternary functions and reduction of gate count. This paper also presents a survey on ternary switching algebra.

**Keywords:** Gate count; Multi-valued logic; Reliability-Unreliability model; Ternary Switching Levels.

## 1. Introduction

Is it time to move beyond zeroes and ones? This is the title of Bernard Cole article's published in 2003 on the official site of the Embedded Development Community [1]. The conclusion is "I think that the economics of semiconductor manufacturing now is forcing us to move beyond zero and one.... Shouldn't we also take another look at multi-valued logic?" This very thought brought many researches to work upon multi-valued logic to bring a new era of technology. This multi-valued logic is focused in this paper recognizing it as a fundamental advancement in circuit technology.

Multi-valued logics are Logical Calculi in which there are more than two truth values. The first known classical logician and father of logic was Aristotle who didn't fully accept the law of excluded middle and admitted that his laws will not apply to future events. Traditionally, in Aristotle's logical calculus, there were only two possible values (i.e., true and false) for any proposition. The later logicians until the coming of the 20th century followed Aristotelian logic, which includes or assumes the law of the excluded middle [2]. The Polish logician and philosopher, Jan Lukasiewicz was a pioneer investigator of multi-valued logic in 1920. The 20th century brought the extension to classical two-valued logic called  $n$ -valued logic for  $n > 2$ .

The most popular in the literature are three-valued logic, the finite-valued with more than 3 values and the infinite-valued (e.g. fuzzy logic) logics. In this paper emphasis is on the three-valued logic, which has three values (i.e., true, false and intermediate) mathematically. The scope of the paper is to implement the novel idea to explore the possibilities and advantages in realizing switching circuits which reduces T-gates. Section 2 covers the preliminaries to ternary logic to recognize and to bring out its importance. In Section 3, ternary switching algebra with the basic operation of T-gates and its conceptual laws and theorems are presented. Design of combinational and sequential circuits with minimum number of multiplexers are presented in section 4. Section 5 gives the conclusion with future work.

## 2. Preliminaries Of Ternary Logic

The ternary logic (also called three-valued or trivalent logic and abbreviated 3VL) is a promising alternative to the conventional binary logic design technique. It is possible for ternary logic to achieve simplicity and energy efficiency in digital design since the logic reduces the complexity of interconnects and chip area, in turn reducing the chip delay [4,5]. It offers better utilization of transmission channels because of the higher information content carried by each line, also gives more efficient error detection and correction codes and possess potentially higher density of information storage. In principle, MVL can provide a means of increasing data processing capability per unit chip area. Furthermore, serial and serial-parallel arithmetic operations can be carried out faster if the ternary logic is employed. One of the main advantages of ternary logic is that it reduces the number of required computation steps. Since each signal can have three distinct values, the number of digits required in a ternary family is  $\log_3 2$  times less than that required in binary logic. It is assumed that ternary-logic

elements can operate at a speed approaching that of the corresponding binary-logic elements. However, if the ternary and binary logic gates are used to take advantage of their respective merits, performance could be significantly improved because ternary logic gates are good candidate for decoding block since it requires less number of gates while binary logic gates are a good candidate for fast computation modules. Thus, ternary design technique combined with the conventional binary logic gate design technique also provides an excellent speed and power consumption characteristics in data path circuit such as full adder and multiplier.

In 1964, Alexander showed that natural base ( $e = 2.71828$ ) is the most efficient radix for implementation of switching circuits. But it was shown that the most efficient radix is 3 compared to 2. To determine the efficiency of base, an estimate is required for the amount of circuitry.

Let us assume a mixed number,  $M = \sum_{i=-f}^n n_i r^i$ , where  $r$  is the base,  $n$  is number of digital positions for fixed values of  $M$ , which is quite realistic for existing types of digital circuitry. The amount of circuitry is directly proportional to the base. The total number of circuits denoted by  $N$  is expressed as  $N = K_1 r^n$ , where  $K_1$  is proportionality constant. With  $n$  digit positions, the number of different numbers which we can represent is  $M = r^n$ . Taking logarithms,  $\ln M = \ln r^n = n(\ln r) = K_2$  where  $K_2$  is constant since  $M$  is a constant. This gives

$$n = \frac{K_2}{\ln r}. \text{ Thus, } N = K_1 \frac{K_2}{\ln r} r = K_3 \frac{r}{\ln r}. \text{ To find the best base we differentiate by } r, \text{ giving}$$

$$\frac{dN}{dr} = \frac{K_3 \left[ (\ln r) - r \cdot \frac{1}{r} \right]}{(\ln r)^2}$$

Setting derivative to zero gives  $\frac{K_3 [(\ln r) - 1]}{(\ln r)^2} = 0$  so that  $(\ln r) - 1 = 0$  or  $\ln r = 1$ . Thus, the most efficient base  $r = e = 2.71828182$ , where  $e$  is called Euler constant [6]. Similarly,  $r = e^2 = 7.389056099$  is more advantageous and most often used base in Electronic computers with digits 0 and 1. The base  $r = e^3 = 20.08553692$  is more efficient. As the value of radix increases, the information carrying capacity of each connection also increases. This accounts to say that multi-valued logical systems, for instance, a three-valued (radix 3) digital realization would be more appropriate than binary.

In real life situations, there is no clear cut binary yes/no requirement. Situations such as yes/no/may be, open/close/half open or half close, up/down/straight or left/right/straight are encountered in the real world scenario. On the other hand, there is always an uncertainty in deciding these values, rather, one would come across situations wherein one has to accept multiple decision values like to what extent the decision may be true and to what extent the decision may be false [7]. Mathematically, Reliability-Unreliability model as shown in Fig.1 implies that the reliability will be interpreted with the boundaries  $0 \leq R \leq 1$ . If  $R=1$ , it is interpreted to be absolutely true and if  $R=0$ , it is interpreted to be absolutely false. If  $R$  being in the vicinity of 0.5 could be interpreted as neither true nor false and above 0.5 is said to be reliability level and below 0.5 is said to be unreliability level.

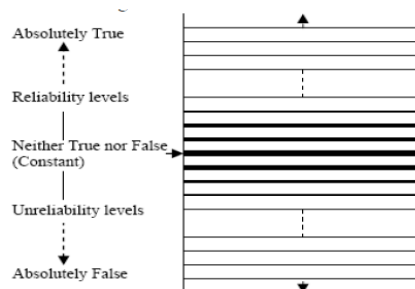


Fig.1. Truth values Reliability-Unreliability model

Expanding the existing logic levels to ternary and higher levels as shown in Fig.2, higher processing rates could be achieved in various applications like memory management, communication throughput and domain specific computation. An evident advantage of a ternary representation over binary is economy of digits. To represent a number in binary system, one needs 58% more digits than that of ternary. Ternary representation admits sign convention also. This is the reason why ternary is casting its applications in the field of Fuzzy logic, Machine Learning, Artificial Intelligence, Data Mining, Robotics, Digital signal processing, Digital control systems and Image Processing. It is mainly applied in error correction, New transforms for encoding and compression, State Assignment, Representation of discrete information, New types of decision diagrams,

Generalized Algebra, Automatic Theorem proving and in automatic telephony. These benefits have shown to be useful in the design of ternary computers, for digital filtering [8].

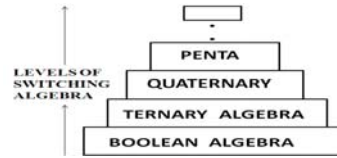


Fig.2. Levels of switching Algebra

### 3. Ternary Switching Algebra

Let a system be  $L$  whose elements called propositions or statements are valued in the set  $\{0, 1, 2\}$  which is denoted by  $Z_3$ . If  $X$  is a proposition, the value of  $X$  can be seen as a mapping  $V : L \rightarrow \{0, 1, 2\}$  such that

$$V(X) = \begin{cases} 2 & \text{if } X \text{ is true} \\ 1 & \text{if } X \text{ is intermediate} \\ 0 & \text{if } X \text{ is false} \end{cases}$$

Ternary has the logic levels '0' corresponding to logic-0 in binary (also called zero element or low voltage), '1' corresponding to an intermediate stage (also called meta stable state) and '2' corresponds to logic-1 in binary (also called universal element or high voltage). The intermediate state can be metaphorically thought of as either true or false. The binary logic is limited to only two states '1' and '0', where as MVL is a set of finite or infinite number of values. In a standard CMOS process, the three supply voltages are vdd, vdd/2 and ground.

Ternary logic gates are the basic building blocks in realizing combinational and sequential logic functions. The implementation is based around (bipolar transistors, MOSFETs etc.) a basic switching elements, which is referred to as T-Gates [9]. The Ternary gate called T-gate qualifies as a universal element in several different sense. Firstly, it should be logically complete with simple operation. Secondly, it should be easily implemented with its straightforward construction. Thirdly, it should possess two essential elements that must be embodied in any logic gate, namely, logic-value thresholding and logic-signal connection of switching [10]. This functional completeness of T-gate is the property of a set of compositions which enables one to synthesize any arbitrary switching function within a particular class. There are several algebras available for the design of ternary switching functions among which, the Post and the Modular algebra have the advantages of similarity with ordinary algebra.

In system  $L$ , a set of operators namely unary and binary are defined. For  $x, y, z \in L$ , there exists an equivalence ( $=$ ) operation [11,12], such that

$$x = x$$

$$\text{If } x = y, \text{ then } y = x$$

$$\text{If } x = y \text{ and } y = z, \text{ then } x = z$$

A general ternary inverter (GTI) is a basic unary operator with one input  $x$  and three outputs. Therefore, the implementation of ternary inverter requires three inverters namely negative ternary inverter (NTI), standard or simple ternary inverter (STI) and positive ternary inverter (PTI) forming an operator set that is complete in logic sense. This basic ternary inverter is used for constructing ternary AND/NAND, ternary OR/NOR etc. They are represented and tabulated as shown in Table I and Fig.3.

$$STI = \overline{X}^1 = 2 - X \quad (1)$$

$$PTI, NTI = \overline{X}^i = \begin{cases} i & \text{if } X \neq i \\ 2 - i & \text{if } X = i \end{cases} \quad (2)$$

where  $i$  takes the value of 2 for PTI and 0 for the NTI operator. The minus sign represents arithmetic subtraction.

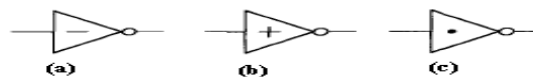


Fig.3. Symbols of Ternary Inverter (a) NTI (b) PTI (c) STI

The unary operator like inverter is a function of  $f : Z_3 \rightarrow Z_3$  and a binary operator like min and max has its ternary extension of  $f : Z_3^2 \rightarrow Z_3^2$ . In general, ternary logical function maps from  $f : Z_3^n \rightarrow Z_3$  and has  $3^{3^n}$  modal functions and  $3^n$  combinations.

Table 1. Function Table For Ternary Inverter

X	$\overline{X^0}$ (NTI)	$\overline{X^2}$ (PTI)	$\overline{X^1}$ (STI)
0	2	2	2
1	0	2	1
2	0	0	0

When  $n = 1$ , one-variable functions  $f(x)$  exist with  $3^{3^1} = 27$  modal functions called Literals as shown in Table II. Similarly, there are  $3^{3^2} = 19683$  two-variable functions and  $3^{3^3} = 76255974849$  three-valued functions [13,14]. Literal is denoted by  $X^{a_i}$ , where  $a_i = 0,1,2,01,02$  and 12 and is defined as given below:

$$X^i = \begin{cases} 0 & \text{if } X \neq i \\ 2 & \text{if } X = i \end{cases} \quad \text{Where } i = 0,1,2 \quad (3)$$

$$X^{01} = X^0 + X^1 \quad (4)$$

$$X^{12} = X^1 + X^2 \quad (5)$$

$$X^{02} = X^0 + X^2 \quad (6)$$

$$X^{01} \bullet X^{12} = X^1 \quad (7)$$

$$X^{01} \bullet X^{02} = X^0 \quad (8)$$

$$X^{02} \bullet X^{12} = X^2 \quad (9)$$

$$X^0 + X^1 + X^2 = 2 \quad (10)$$

Table 2. Function Table Of Unary Functions

X	$X^0$	$X^1$	$X^2$	$X^{01}$	$X^{12}$	$X^{02}$
0	2	0	0	2	0	2
1	0	2	0	2	2	0
2	0	0	2	0	2	2

It can be proved that the complement or negation of literals ( $X^i$ ) give the following observed Eq.(12-15) which are helpful in reduction of ternary gates during implementation. The negation is defined and represented as shown in Fig.4 and Eq.(11).

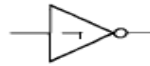


Fig.4. Symbol for Negation

$$COM(X^i) \text{ or } NEQ(X^i) = \overline{X^i} = \begin{cases} 0 & \text{if } X=i \\ 2 & \text{if } X \neq i \end{cases} \quad (11)$$

$$X^2 = \overrightarrow{X^{01}} \text{ \& } X^{01} = \overrightarrow{X^2} \quad (12)$$

$$X^1 = \overrightarrow{X^{02}} \text{ \& } X^{02} = \overrightarrow{X^1} \quad (13)$$

$$X^0 = \overrightarrow{X^{12}} \text{ \& } X^{12} = \overrightarrow{X^0} \quad (14)$$

$$\overrightarrow{0} = 2 \text{ \& } \overrightarrow{2} = 0 \quad (15)$$

This observed result is used to show the reduction in gate count and also in simplification of ternary function.

The operation of addition (+) and multiplication (.) on L, which can be called Ternary OR (TOR) and Ternary AND (TAND) respectively, represent two multiple input operators. It is represented by following equations and tabulated as shown in Table 3 and Fig.5 and 6.

Logic Sum or TOR:

$$X1 + X2 + \dots + Xn = \text{MAX}(X1, X2, \dots, Xn) \quad (16)$$

Logic Product or TAND:

$$X1 \bullet X2 \bullet \dots \bullet Xn = \text{MIN}(X1, X2, \dots, Xn) \quad (17)$$

Similarly, TNAND is

$$\overline{X1 \bullet X2 \bullet \dots \bullet Xn} = \text{MIN}(\overline{X1 \bullet X2 \bullet \dots \bullet Xn}) \quad (18)$$

TNOR is

$$\overline{X1 + X2 + \dots + Xn} = \text{MAX}(\overline{X1 + X2 + \dots + Xn}) \quad (19)$$

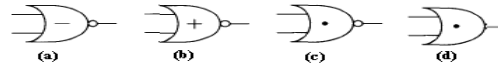


Fig.5. Symbols of MAX operator  
(a)NTNOR (b)PTNOR (c)STNOR and (d)TOR

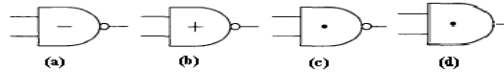


Fig.6.Symbols of MIN operator  
(a)NTNAND (b)PTNAND (c)STNAND and (d)TAND

Clearly  $(L, +, \cdot)$  is a distributive lattice with zero element(0) and universal element(2) [15-18]. Thus the following laws hold for any  $x, y, z \in L$ :

Idempotent:  $X + X = X$   $X \cdot X = X$

Commutative:  $X + Y = Y + X$

$X \cdot Y = Y \cdot X$

Associative:  $(X + Y) + Z = X + (Y + Z)$   
 $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$

Absorption:  $X + X \cdot Y = X$   $X \cdot (X + Y) = X$

Distributive:  $X + Y \cdot Z = (X + Y) \cdot (X + Z)$   
 $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$

It is evident that laws of identity elements, holds here.

$$X + 0 = X \quad (20)$$

$$X \cdot 0 = 0 \quad (21)$$

$$X + 2 = 2 \quad (22)$$

$$X \cdot 2 = X \quad (23)$$

$$X \cdot 1 = 1 \text{ (for cases } X \neq 0) \quad (24)$$

$$X + 1 = 1 \text{ (for cases } X \neq 2) \text{ \& } 2 \text{ (for } x=2) \quad (25)$$

DeMorgan's Theorem holds for ternary logic when the three types of inverters are used.

$$\overline{(X + Y)}^0 = \overline{X}^0 \cdot \overline{Y}^0 \quad (26)$$

$$\overline{(X \cdot Y)}^0 = \overline{X}^0 + \overline{Y}^0 \quad (27)$$

$$\overline{(X + Y)}^1 = \overline{X}^1 \cdot \overline{Y}^1 \quad (28)$$

$$\overline{(X \cdot Y)}^1 = \overline{X}^1 + \overline{Y}^1 \quad (29)$$

$$\overline{(X + Y)}^2 = \overline{X}^2 \cdot \overline{Y}^2 \quad (30)$$

$$\overline{(X \cdot Y)}^2 = \overline{X}^2 + \overline{Y}^2 \quad (31)$$

$$\overline{\overline{X}^1}^1 = X \quad (32)$$

Ternary Ex-OR function is mod-3 addition of ternary numbers. Modulo-3 sum is the sum of two integers ignoring the carry digits in the addition. Modulo-3 addition is an important function, since so many redundant code techniques use half-adding functions [19,20]. It is denoted and expressed as given in Fig.7 and Table 4.

$$X \oplus Y = MODSUM(X, Y) = (X + Y) \bmod 3 \quad (33)$$

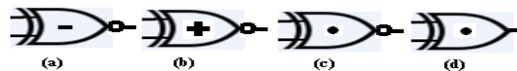


Fig.7. Symbols of MODULO SUM operator  
(a)NTEQV (b)PTEQV (c)STEQV and (d)TEXOR

Ternary functions of one or more variables may be represented in truth table or K-map form or algebraically in canonical form as a product of sum or sum of product. According to Expansion theorem [21], any ternary function  $f(X_1, X_2, \dots, X_n)$  may be generated from  $(X_1, X_2, \dots, X_n)$  by means of  $(+)$ ,  $(\cdot)$  and the unary functions  $X^0, X^1, X^2$  as given below:

$$f(X_1, X_2, \dots, X_n) = 2 \cdot F_2(X_1, X_2, \dots, X_n) + 1 \cdot F_1(X_1, X_2, \dots, X_n) + 0 \cdot F_0(X_1, X_2, \dots, X_n)$$

$$\text{i.e., } f = 2 \cdot F_2 + 1 \cdot F_1 + 0 \cdot F_0 \quad (34)$$

where  $F_k$  equals 2, when value of the function  $f$  equals  $k$ , otherwise, it is 0. Applying equations (21) and (23) to the above equation, the function may be represented by

$f = F_2 + 1 \cdot F_1$  for canonical Sum of Product form and  $f = F_2 \cdot (1 + F_1)$  for canonical Product of Sum form.

#### 4. Design Of Ternary Circuits

An approach for implementing ternary function is to convert given ternary variable into unary variable using ternary to unary decoder as shown in Fig.8. The other circuit design and concepts can also be found in reference [22-27].

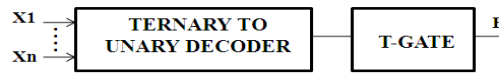


Fig.8. Implementation of ternary function

##### I. Design of decoder and obtaining literals

To design ternary circuits, we start with the design of decoder which is a basic building block as shown in Fig.9, which operates according to the Table 2.

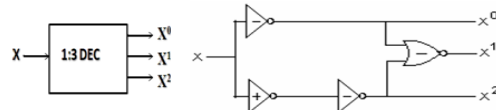


Fig.9. Block diagram and Circuit diagram of 1x3 Decoder

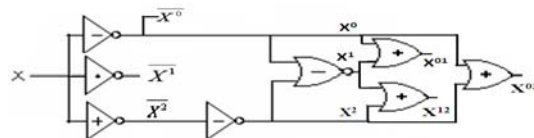


Fig.10. Circuit for obtaining Unary functions

A decoder is a combinational circuit that converts the ternary information from  $n$  input lines to  $3^n$  unique output lines Design to obtain the literals is also shown in Fig.10.

##### II. Design of multiplexer

A ternary multiplexer is a combinational circuit that selects one of the  $3^n$  input lines based, on a set of  $n$  selection lines and directs it to a single output line. The design of  $3 \times 1$  multiplexer (MUX) is as presented in Fig.11 and operates as given in Table 5. In this paper,  $3 \times 1$  MUX is taken as a basic building block to explore the realization of  $9 \times 1$  MUX,  $27 \times 1$  MUX, Half Adder, Half Subtractor, Full Adder, Full Subtractor, Multiplier, 1-bit and 2-bit Comparator, 1-bit and 2-bit Position shifter and Barrel shifter.

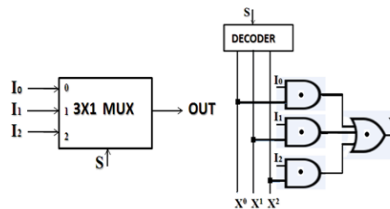


Fig.11. Block diagram and circuit diagram of  $3 \times 1$  MUX

##### III. Design of $9 \times 1$ MUX using $3 \times 1$ MUX

A  $9 \times 1$  MUX is built using four  $3 \times 1$  MUX as shown in Fig.12 and functions as given in Table 6. A  $9 \times 1$  MUX selects one among the 9 inputs based on 2 select lines.

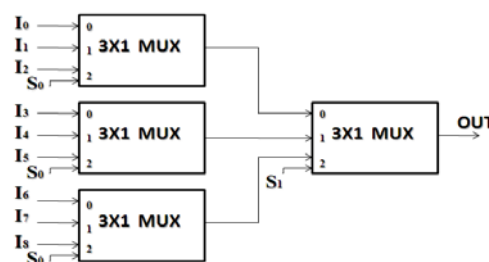


Fig.12. Block diagram of  $9 \times 1$  MUX

#### IV. Design of Half Adder using 3x1 MUX

A ternary half adder (HA) is a circuit that adds two bits and generates a sum and carry using Modulo-3 addition. There are two inputs and two outputs and, consequently, two decoders are required. The half adder functions as shown in Table 7 can be realized as given in Fig.13(a-c) using 3x1 MUX.

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

Fig.13(a). K-maps for Half Adder

$$f_s = A^0 B^2 + A^1 B^1 + A^2 B^0 + 1 \cdot (A^0 B^1 + A^1 B^0 + A^2 B^2) \quad (35)$$

$$f_c = 1 \cdot (A^2 B^1 + A^1 B^2) = 1 \cdot (A^2 \overline{B^0} + A^0 \overline{B^2}) \quad (36)$$

The gate count for half adder is  $f_s + f_c = 9 + 6 = 15$  but we have reduced the gate count to  $9 + 4 = 13$  using negation.

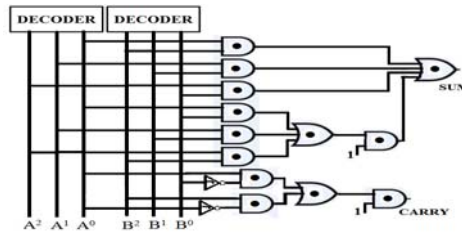


Fig.13(b). Circuit diagram of Half Adder

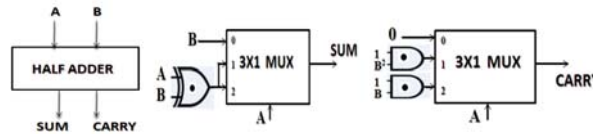


Fig.13(c). Block diagram of half adder using 3x1 MUX

#### V. Design of Half Subtractor using 3x1 MUX

Ternary half subtractor (HS) is a circuit that will subtract one from the other number (i.e., A-B) and generate a difference and borrow using ternary logic. The half subtractor function is as shown in Table 8 and can be realized as given in Fig.14(a-c) using 3x1 mux.

$$f_d = A^0 B^1 + A^1 B^2 + A^2 B^0 + 1 \cdot (A^1 B^0 + A^2 B^1 + A^0 B^2) \quad (37)$$

$$f_b = 1 \cdot (A^0 B^1 + A^1 B^2) = 1 \cdot (A^0 \overline{B^0} + A^1 \overline{B^2}) \quad (38)$$

The gate count for half subtractor is  $f_d + f_b = 9 + 6 = 15$  but using negation we reduced the gate count to  $9 + 4 = 13$ .

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

Fig.14(a). K-maps for Half Subtractor

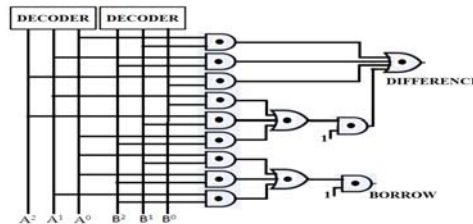


Fig.14(b). Circuit for Half Subtractor using 3x1 MUX

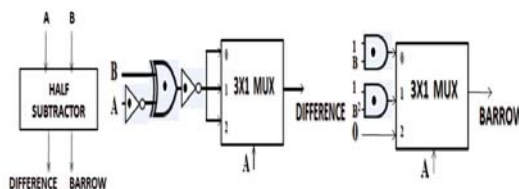


Fig.14(c). Block diagram for Half Subtractor using 3x1 MUX

## VI. Design of Full Adder using 3x1 MUX

A full adder (FA) is a circuit that will add three bits and generates a sum and a carry. Fig.15(a-b) shows the design of full adder realization with 3x1 mux functioning according to Table 9.

$$f_s = A^0 B^2 C^0 + A^1 B^1 C^0 + A^2 B^0 C^0 + A^1 B^0 C^1 + A^0 B^1 C^1 + A^2 B^2 C^1 + A^0 B^0 C^2 + A^2 B^1 C^2 + A^1 B^2 C^2 + A^2 B^2 C^0 + A^0 B^0 C^1 + A^2 B^1 C^1 + A^1 B^2 C^1 + A^2 B^0 C^2 + A^1 B^1 C^2 + A^0 B^2 C^2 \quad (39)$$

$$f_c = A^2 B^2 C^2 + 1.(B^1 C^2 + B^2 C^1 + A^2 C^1 + A^1 C^2 + A^2 B^1 C^0 + A^2 B^2 C^0 + A^1 B^2 C^0 + A^1 B^1 C^1) = A^2 B^2 C^2 + 1.(B^0 C^2 + B^2 C^0 + A^2 C^0 + A^0 C^2 + A^2 B^1 C^0 + A^2 B^2 C^0 + A^1 B^2 C^0 + A^1 B^1 C^1) \quad (40)$$

The gate count for full adder is  $f_s + f_c = 21 + 11 = 32$  but using negation we reduced the gate count to  $21 + 15 = 36$ .

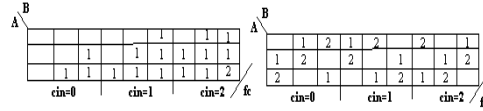


Fig.15(a). K-maps for Full Adder

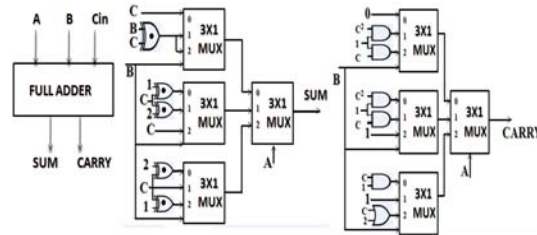


Fig.15(b). Block diagram of Full Adder using 3x1 MUX

## VII. Design of Full Subtractor using 3x1 MUX

Full subtractor (FS) is a circuit that will subtract three bits (i.e.,  $(A - (B - \text{Bin}))$ ), and generates a difference and a borrow. Fig.16(a-b) shows the design of full subtractor realization with 3x1 mux functioning according to truth table in Table 10.

$$f_d = A^0 B^0 C^1 + A^0 B^1 C^0 + A^2 B^2 C^0 + A^1 B^0 C^2 + A^1 B^1 C^1 + A^1 B^2 C^0 + A^2 B^0 C^0 + A^2 B^1 C^2 + A^2 B^2 C^1 + A^0 B^0 C^0 + A^1 B^1 C^2 + A^1 B^2 C^1 + A^2 B^0 C^1 + A^2 B^1 C^0 + A^2 B^2 C^2 \quad (41)$$

$$f_b = A^0 B^2 C^2 + 1.(A^0 C^2 + A^0 C^1 + A^0 B^2 + B^2 C^1 + B^1 C^2 + A^0 B^1 C^0 + A^1 B^1 C^1) = A^0 B^2 C^2 + 1.(A^2 C^2 + A^0 C^0 + A^2 B^2 + B^2 C^0 + B^1 C^2 + A^0 B^1 C^0 + A^1 B^1 C^1) \quad (42)$$

The gate count for full subtractor is  $f_d + f_b = 21 + 15 = 36$  but using negation we have reduced the gate count to  $21 + 11 = 32$ .

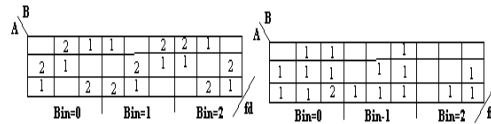


Fig.16(a). K-maps for Full Subtractor

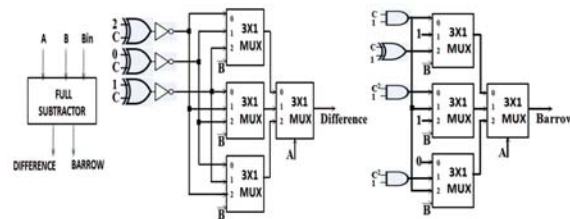


Fig.16(b). Block diagram and circuit diagram of Full Subtractor using 3x1 MUX

## VIII. Design of 1-bit Multiplier using 3x1 MUX

A multiplier multiplies two bits and generates the product. Truth table for multiplier is shown in Table 11 and Fig.17(a-b). The gate count of multiplier is  $f_p + f_c = 7 + 2 = 9$ .

$$f_p = A^2 B^1 + A^1 B^2 + 1.(A^1 B^1 + A^2 B^2) \quad (43)$$

$$f_c = 1.(A^2 B^2) \quad (44)$$



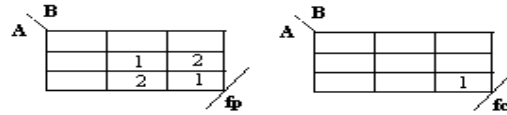


Fig.17(a). K-maps for Multiplier

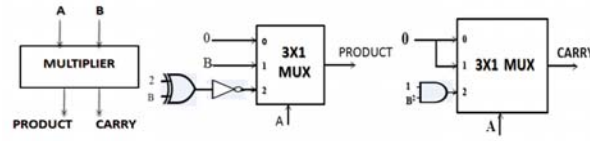


Fig.17(b). Block diagram of Multiplier

### IX. Design of 1-bit comparator using 3x1 MUX

A magnitude comparator is a combinational circuit that compares two bits A & B and determines their relative magnitudes. The comparison of two bits is an operation that determines if one number is greater than, less than or equal to other number as Fig.18(a-b) and Table 12 shows the design of 1-bit comparator which gives  $Y=f(A>B)$  when  $en=0$ ,  $Y=f(A=B)$  when  $en=1$  and  $Y=f(A<B)$  when  $en=2$ .

$$f(A>B)=A^1B^0+A^2B^0+A^2B^1=\overline{A^0}B^0+A^2B^1 \quad (45)$$

$$f(A=B)=A^0B^0+A^1B^1+A^2B^2 \quad (46)$$

$$f(A<B)=A^0B^1+A^0B^2+A^1B^2=\overline{A^0}B^1+\overline{A^0}B^2+A^1B^2 \quad (47)$$

The gate count for comparator is  $4+4+4=12$  but using negation we reduced the gate count to  $3+4+3=10$ .

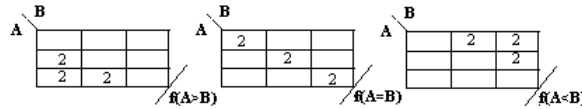


Fig.18(a). K-maps for 1-bit Comparator

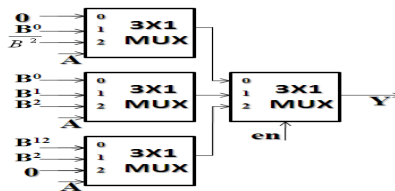


Fig.18(b). Block diagram of 1-bit Comparator

### X. Design of 2-bit comparator using 27x1 MUX

The design of 27x1 MUX is designed as building block for the implementation of 2-bit comparator as shown in Fig.19 and Table 13.

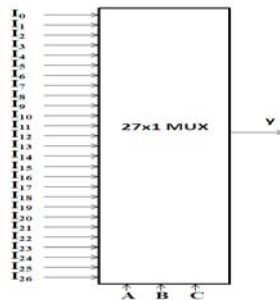


Fig.19. Block diagram of 27x1 MUX

The design of 2-bit Comparator is given in the Fig.20(a-d) and Table.XIV which needs three 27x1 MUX and one 3x1 MUX which selects one of the function. If  $en=0$ ,  $Y=f(A_1A_0>B_1B_0)$ , if  $en=1$ ,  $Y=f(A_1A_0=B_1B_0)$  and if  $en=2$ ,  $Y=f(A_1A_0<B_1B_0)$ .

$$f(A>B)=A_1^1B_1^0+A_1^2B_1^0+A_1^2B_1^1+A_1^2A_0^2B_0^0+A_1^0B_1^0B_0^0+A_1^2A_0^1B_0^0+A_1^2B_1^0B_0^0+A_1^0B_1^0B_0^0+A_1^1A_0^1B_1^0B_0^0+A_1^1A_0^2B_1^0B_0^0+A_1^1A_0^2B_1^1B_0^0+A_1^1A_0^2B_1^1B_0^1+A_1^1A_0^2B_1^1B_0^2+A_1^1A_0^2B_1^1B_0^3+A_1^1A_0^2B_1^1B_0^4+A_1^1A_0^2B_1^1B_0^5+A_1^1A_0^2B_1^1B_0^6+A_1^1A_0^2B_1^1B_0^7+A_1^1A_0^2B_1^1B_0^8+A_1^1A_0^2B_1^1B_0^9+A_1^1A_0^2B_1^1B_0^{10}+A_1^1A_0^2B_1^1B_0^{11}+A_1^1A_0^2B_1^1B_0^{12}+A_1^1A_0^2B_1^1B_0^{13}+A_1^1A_0^2B_1^1B_0^{14}+A_1^1A_0^2B_1^1B_0^{15}+A_1^1A_0^2B_1^1B_0^{16}+A_1^1A_0^2B_1^1B_0^{17}+A_1^1A_0^2B_1^1B_0^{18}+A_1^1A_0^2B_1^1B_0^{19}+A_1^1A_0^2B_1^1B_0^{20}+A_1^1A_0^2B_1^1B_0^{21}+A_1^1A_0^2B_1^1B_0^{22}+A_1^1A_0^2B_1^1B_0^{23}+A_1^1A_0^2B_1^1B_0^{24}+A_1^1A_0^2B_1^1B_0^{25}+A_1^1A_0^2B_1^1B_0^{26} \quad (48)$$

$$f(A=B) = A_1^0 A_0^0 B_1^0 B_0^0 + A_1^0 A_0^0 B_1^1 B_0^0 + A_1^0 A_0^0 B_1^2 B_0^0 + A_1^1 A_0^0 B_1^0 B_0^0 + A_1^1 A_0^0 B_1^1 B_0^0 + A_1^1 A_0^0 B_1^2 B_0^0 + A_1^2 A_0^0 B_1^0 B_0^0 + A_1^2 A_0^0 B_1^1 B_0^0 + A_1^2 A_0^0 B_1^2 B_0^0 = [A_1^0 B_1^0 + A_1^1 B_1^1 + A_1^2 B_1^2] \cdot [A_0^0 B_0^0 + A_0^1 B_0^1 + A_0^2 B_0^2] \quad (49)$$

$$f(A < B) = A_1^0 B_1^1 + A_1^0 B_1^2 + A_1^1 B_1^2 + A_1^0 A_0^0 B_0^1 + A_1^0 A_0^0 B_0^2 + A_1^0 A_0^1 B_0^0 + A_1^0 A_0^1 B_0^1 + A_1^0 A_0^1 B_0^2 + A_1^0 A_0^2 B_0^0 + A_1^0 A_0^2 B_0^1 + A_1^0 A_0^2 B_0^2 + A_1^1 A_0^0 B_0^0 + A_1^1 A_0^0 B_0^1 + A_1^1 A_0^0 B_0^2 + A_1^1 A_0^1 B_0^0 + A_1^1 A_0^1 B_0^1 + A_1^1 A_0^1 B_0^2 + A_1^1 A_0^2 B_0^0 + A_1^1 A_0^2 B_0^1 + A_1^1 A_0^2 B_0^2 + A_1^2 A_0^0 B_0^0 + A_1^2 A_0^0 B_0^1 + A_1^2 A_0^0 B_0^2 + A_1^2 A_0^1 B_0^0 + A_1^2 A_0^1 B_0^1 + A_1^2 A_0^1 B_0^2 + A_1^2 A_0^2 B_0^0 + A_1^2 A_0^2 B_0^1 + A_1^2 A_0^2 B_0^2 \quad (50)$$

The gate count of 2-bit comparator is 13+9+13=35

we reduced the gate count to 8+9+9=26.

B1B0																	
A1A0	00	01	02	10	11	12	20	21	22								
		0	1	2	3	4	5	6	7								
00	2																
01	2	2															
02	2	2	2														
10	2	2	2	2													
11	2	2	2	2	2												
12	2	2	2	2	2	2											
20	2	2	2	2	2	2	2										
21	2	2	2	2	2	2	2	2									
22	2	2	2	2	2	2	2	2	2								

Fig.20(a).K-maps for F(A>B)

B1B0																	
A1A0	00	01	02	10	11	12	20	21	22								
		0	1	2	3	4	5	6	7								
00	2																
01	2	2															
02			2														
10			2	2													
11					2												
12					2	2											
20						2	2										
21						2	2	2									
22						2	2	2	2								

Fig.20(b).K-maps for F(A=B)

B1B0																	
A1A0	00	01	02	10	11	12	20	21	22								
		0	1	2	3	4	5	6	7								
00	2	2	2	2	2	2	2	2	2								
01		2	2	2	2	2	2	2	2								
02			2	2	2	2	2	2	2								
10				2	2	2	2	2	2								
11					2	2	2	2	2								
12						2	2	2	2								
20							2	2	2								
21								2	2								
22									2								

Fig.20(c).K-maps for F(A<B)

## XI. Design of 1-bit & 2-bit position shifter and Barrel Shifter

The shifter circuit is as designed in Fig.21 and Table 15 which shifts the bits of an n input vector by 1-bit position to the right. It fills the vacant position on the left side with zero. If S=0, the input is loaded as output which is said to work in parallel mode, if S=1, the input is shifted by one bit position and if S=2, the input is shifted by two-bit positions padded by zeros in the left position. The more versatile shifter circuit will be able to shift by more bit positions at a time. If the bits that are shifted out are placed into the vacated positions on the left, then the circuit effectively rotates the bits of the input vector by a specified number of bit positions. Such a circuit is often called a barrel shifter. Barrel shifter works as functioned in the Table 16 and Fig.21(b). The shifter takes the parallel load for S=0. And for S=1, the output is shifted by one position and values are rotated. Similarly for S=2, the output is shifted by two bit position and rotated circularly.

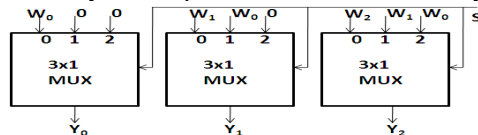


Fig.21(a). Block diagram of 1 & 2 bit Position Shifter

$$Y_0 = s^0 w_0 \quad (51)$$

$$Y_1 = s^0 w_1 + s^1 w_0 \quad (52)$$

$$Y_2 = s^0 w_2 + s^1 w_1 + s^2 w_0 \quad (53)$$

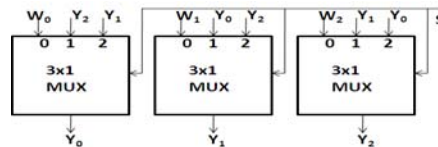


Fig.21(b). Block diagram of Barrel Shifter

$$Y_0 = s^0 w_0 + s^1 Y_2 + s^2 Y_1 \quad (54)$$

$$Y_1 = s^0 w_1 + s^1 Y_0 + s^2 Y_2 \quad (55)$$

$$Y_2 = s^0 w_2 + s^1 Y_1 + s^2 Y_0 \quad (56)$$

## 5. Conclusion

The ternary logic is a promising alternative to the conventional binary logic design technique. The ternary and binary logic gates can be used to take advantage of their respective merits, to improve performance in terms of computation speed and power consumption. Expanding the existing logic levels to higher levels higher processing rates could be achieved. In this paper, we have implemented a technique to reduce the gate count.

## Acknowledgements

The authors would like to thank Mr. Sathish K. Manocha, Dept., of Electronics and Communication for his valuable guidance.

## References

- [1] [http://www.embedded.com/columns/netcentricview/13100886?\\_requestid=133447](http://www.embedded.com/columns/netcentricview/13100886?_requestid=133447)
- [2] [http://en.wikipedia.org/wiki/Ternary\\_logic](http://en.wikipedia.org/wiki/Ternary_logic)
- [3] K. C. Smith. (1981): The Prospects for Multivalued Logic: A Technology and Applications View, *IEEE Transactions on Computers*, Vol. C-30, Issue.9, pp.619-634.
- [4] Sheng Lin et al. (2009): CNTFET-Based Design of Ternary Logic Gates and Arithmetic Circuits, *IEEE Trans. on Nanotechnology*, Vol. PP, Issue.99, pp.1-1.
- [5] S.L.Hurst. (1984): Multivalued logic - Its status and its future, *IEEE Trans. on Computers*, vol. C-33, pp. 1160-1179.
- [6] Raymond E.Miller. (1966): Switching Theory, Vol. I, John Wiley & Sons, pp.8-9.
- [7] D. Venkat Reddy et. al (2008): Sequential Circuits In The Framework Of  $(2n+1)$ -ary Discrete Logic, *IJCSNS International Journal of Computer Science and Network Security*, Vol.8 No.7, July, pp.175-181.
- [8] Marek Perkowski (2006): Introduction to multivalued logic.Avaliable as: <http://web.cecs.pdx.edu/~mperkows/temp/JULY/2006.Introduction-to-MV-logic.ppt>
- [9] A.P. Dhande et. al. (2005): Design And Implementation Of 2 Bit Ternary ALU Slice, 3rd International Conference, Sciences of Electronic (SETIT), *IEEE Transc.*, pp.1-11.
- [10] K.C. Smith. (1988): Multiple-Valued Logic, A Tutorial and Appreciation, Survey & Tutorial Series, *IEEE Transc. in computers*, Vol.21, Issue.4, pp.17-27.
- [11] H.T. Mouftah. (1975): Three-valued logic and its implementation with COS/MOS integrated circuits, D.Sc. thesis, Laval University, Quebec, Canada.
- [12] H.T. Mouftah. (1975): A Study On The Implementation Of Three-valued Logic, University of Toronto, Toronto, Ontario, Canada, pp.123-126.
- [13] M. Yoeli et. al. (1965): Logical Design of Ternary Switching Circuits, *IEEE Transactions on Electronic Computers*, Vol.EC-14, Issue.1, pp.19-29.
- [14] Jorge Pedraza Arpasi (2003): A Brief Introduction to Ternary Logic, pp.1-13.
- [15] X.W. Wu. (1990): CMOS ternary logic circuits, *IEE Proceedings*, Vol. 137, Pt. G, No. 1, pp.21-27.
- [16] Stephen. et. al. (1972): The Relationship Between Multivalued Switching Algebra and Boolean Algebra Under Different Definitions of Complement, *IEEE Transactions on Computers*, VOL. C-21, Issue. 5, pp.479-485.
- [17] Chung-Yu Wu. et. al. (1993): Design and Application of Pipelined Dynamic CMOS Ternary Logic and Simple Ternary Differential Logic, *IEEE Journal Of Solid-state Circuits*, VOL. 28. Issue.8, pp.895-906.
- [18] Lee-Ju Choi et. al.(1990): Design And Implementation Of Ternary Exoatmospheric Autopilot System, Digital Avionics System Conference, *IEEE Transc.*, pp.58-63.
- [19] D. I. Porat. (1969): Three-valued digital systems, *PROC. IEE*, Vol. 116, No. 6, pp.947-954.
- [20] R. P. Hallworth. et. al. (1961): Semiconductor Circuits For Ternary Logic, The Institution of Electrical Engineers, Monograph No. 482 E, *IEEE trans.*, Vol. 109, Part C., pp.219-225.
- [21] Robert L.Herrmann. (1968): Selection and implementation of a ternary switching algebra, Proceedings of AFIPS Joint Computer Conference, Spring Joint Computer Conference, pp.283-290.
- [22] I.Halpern. et. al. (1968): Ternary arithmetic unit, *PROC. IEE*, Vol. 115, No. 10, pp.1385-1388.
- [23] Prabhakara C. Balla. et. al. (1984): Low Power Dissipation MOS Ternary Logic Family, *IEEE Journal Of Solid-State Circuits*, Vol. SC-19, No. 5, pp.739-749.
- [24] Mozammed H.A.Khan. (2006): Design of reversible/ quantum ternary multiplexer and demultiplexer, *Engineering letters*, 13:2,EL\_13\_2\_3, Advanced Online Publication:4.
- [25] E. Sipos. et. al. (2008): A Method to Design Ternary Multiplexers Controlled by Ternary Signals Based on SUS-LOC, Proceedings of the IEEE International Conference on Automation, quality and Testing, Robotics, IEEE Computer Society, Vol.3, pp.402-407.
- [26] Stephen Brown and Zvonko vranesic. (2008): Fundamentals Of Digital Logic With Verilog Design, Special Indian Edition, Second edition, Tata McGraw-Hill.
- [27] R. F Tinder. (2000): Engineering digital design, Second edition, Academic press.



**A.Sathish Kumar** received Bachelors in Electronics and Communication Engineering from Sona College of Technology, Salem, Tamilnadu in the year 2009 and currently pursuing Masters in VLSI Design at Amrita School of Engineering, Bangalore, India. He is a member of IETE.



**A. Swetha Priya** received Bachelors in Electronics and Instrumentation Engineering from Amrita School of Engineering, Bangalore in the year 2009 and currently pursuing Masters in VLSI Design at Amrita School of Engineering, Bangalore, India

Table 3. Truth Table For Basic Gates

A	B	TAND	STNAND	PTNAND	NTNAND	TOR	STNOR	PTNOR	NTNOR
0	0	0	2	2	2	0	2	2	2
0	1	0	2	2	2	1	1	2	0
0	2	0	2	2	2	2	0	0	0
1	0	0	2	2	2	1	1	2	0
1	1	1	1	2	0	1	1	2	0
1	2	1	1	2	0	2	0	0	0
2	0	0	2	2	2	2	0	0	0
2	1	1	1	2	0	2	0	0	0
2	2	2	0	0	0	2	0	0	0

Table 4. Operation Table For Modulo Sum

A	B	TXOR	STEQV	PTEQV	NTEQV
0	0	0	2	2	2
0	1	1	1	2	0
0	2	2	0	0	0
1	0	1	1	2	0
1	1	2	0	0	0
1	2	0	2	2	2
2	0	2	0	0	0
2	1	0	2	2	2
2	2	1	1	2	0

Table 5. Operation Table Of 3x1 MUX

S	OUT
0	I0
1	I2
2	I3

Table 6. Function Table Of 9x1 MUX

S1	S0	OUT
0	0	I0
0	1	I1
0	2	I2
1	0	I3
1	1	I4
1	2	I5
2	0	I6

2	1	I7
2	2	I8

Table 7. Function Table Of Half Adder

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
0	2	2	0
1	0	1	0
1	1	2	0
1	2	0	1
2	0	2	0
2	1	0	1
2	2	1	1

Table 8. Function Table Of Half Subtractor

A	B	DIFF	BORR
0	0	0	0
0	1	2	1
0	2	1	1
1	0	1	0
1	1	0	0
1	2	2	1
2	0	2	0
2	1	1	0
2	2	0	0

Table 9. Function Table Of Full Adder

A	B	Cin	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	0	2	2	0
0	1	0	1	0
0	1	1	2	0
0	1	2	0	1
0	2	0	2	0
0	2	1	0	1
0	2	2	1	1
1	0	0	1	0
1	0	1	2	0
1	0	2	0	1
1	1	0	2	0
1	1	1	0	1
1	1	2	1	1
1	2	0	0	1
1	2	1	1	1
1	2	2	2	1
2	0	0	2	0
2	0	1	0	1
2	0	2	1	1
2	1	0	0	1
2	1	1	1	1
2	1	2	2	1
2	2	0	1	1
2	2	1	2	1
2	2	2	0	2

Table 10. Function Table Of Full Subtractor

A	B	Bin	DIFF	BORR
0	0	0	0	0
0	0	1	2	1
0	0	2	1	1
0	1	0	2	1
0	1	1	1	1
0	1	2	0	1
0	2	0	1	1
0	2	1	0	1
0	2	2	2	2
1	0	0	1	0
1	0	1	0	0
1	0	2	2	1
1	1	0	0	0
1	1	1	2	1
1	1	2	1	1
1	2	0	2	1
1	2	1	1	1
1	2	2	0	1
2	0	0	2	0

2	0	1	1	0
2	0	2	0	0
2	1	0	1	0
2	1	1	0	0
2	1	2	2	1
2	2	0	0	0
2	2	1	2	1
2	2	2	1	1

Table 11. Function Table For 1-Bit Multiplier

A	B	PROD	CARRY
0	0	0	0
0	1	0	0
0	2	0	0
1	0	0	0
1	1	1	0
1	2	2	0
2	0	0	0
2	1	2	0
2	2	1	1

Table 12. Function Table For 1-Bit Comparator

A	B	A>B	A=B	A<B
0	0	0	2	0
0	1	0	0	2
0	2	0	0	2
1	0	2	0	0
1	1	0	2	0
1	2	0	0	2
2	0	2	0	0
2	1	2	0	0
2	2	0	2	0

Table 13. Function Table For 27x1 MUX

A	B	C	Y
0	0	0	I0
0	0	1	I1
0	0	2	I2
0	1	0	I3
0	1	1	I4
0	1	2	I5
0	2	0	I6
0	2	1	I7
0	2	2	I8
1	0	0	I9
1	0	1	I10
1	0	2	I11
1	1	0	I12
1	1	1	I13
1	1	2	I14
1	2	0	I15
1	2	1	I16
1	2	2	I17
2	0	0	I18

2	0	1	I19
2	0	2	I20
2	1	0	I21
2	1	1	I22
2	1	2	I23
2	2	0	I24
2	2	1	I25
2	2	2	I26

Table 14. Function Table For 2-Bit Comparator

A1	A0	B1	B0	A>B	A=B	A<B
0	0	0	0	0	2	0
0	0	0	1	0	0	2
0	0	0	2	0	0	2
0	0	1	0	0	0	2
0	0	1	1	0	0	2
0	0	1	2	0	0	2
0	0	2	0	0	0	2
0	0	2	1	0	0	2
0	0	2	2	0	0	2
0	1	0	0	2	0	0
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
2	2	1	2	2	0	0
2	2	2	0	2	0	0
2	2	2	1	2	0	0
2	2	2	2	0	2	0

Table 15. Function Table For 1-Bit & 2-Bit Position Shifter

X	Y0	Y1	Y2
0	W0	W1	W2
1	0	W0	W1
2	0	0	W0

Table 16. Function Table For Barrel Shifter

S	Y0	Y1	Y2
0 (PARALLEL LOAD)	W0	W1	W2
1 (SHIFT BY 1-BIT POSITION TO RIGHT)	W2 W1 W0	W0 W2 W1	W1 W0 W2
2 (SHIFT BY 2-BIT POSITION TO RIGHT)	W1 W2 W0	W2 W0 W1	W0 W1 W2

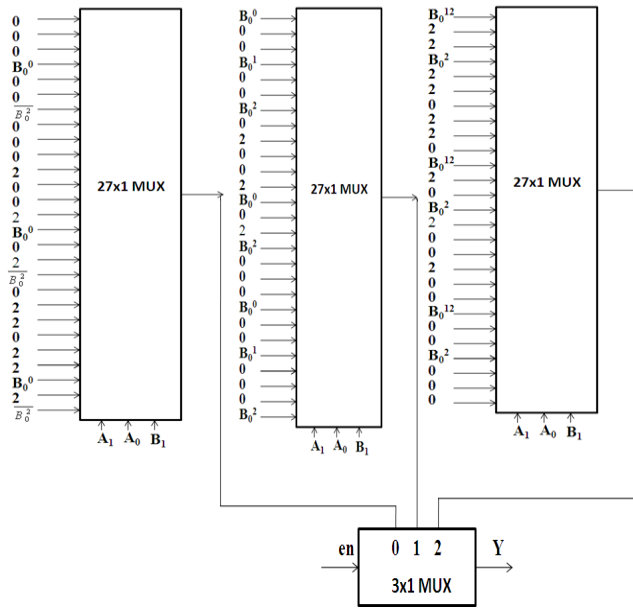


Fig.20(d). Block diagram of 2-bit comparator