

Title: Ternary Digital System: Concepts and Applications

Authors: A P Dhande, V T Ingole, V R Ghiye

Published by SM Online Publishers LLC

Copyright © 2014 SM Online Publishers LLC

ISBN: 978-0-9962745-0-0

All book chapters are Open Access distributed under the Creative Commons Attribution 3.0 license, which allows users to download, copy and build upon published articles even for commercial purposes, as long as the author and publisher are properly credited, which ensures maximum dissemination and a wider impact of the publication. Upon publication of the eBook, authors have the right to republish it, in whole or part, in any publication of which they are the author, and to make other personal use of the work, identifying the original source.

Statements and opinions expressed in the book are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published chapters. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

First published October, 2014

Online Edition available at www.smgebooks.com

For reprints, please contact us at ebooks@esciencemedicine.com

Combinational Logic Design

INTRODUCTION

A combinational circuit consists of input variables, ternary logic gates and output variables. The output of the circuit depends only upon the present input. Logic gates accept signals from the input variables and generate output signals. This process transforms ternary information from the given input data to the required ternary output data. Figure 5.1 shows the block diagram of a combinational circuit.

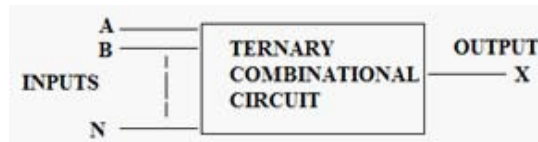


Figure 5.1: Ternary combinational circuit.

For the operation of ternary circuit, an additional ternary to unary block converter is essential to convert ternary input to its unary values, describe in the next section.

In this chapter design examples using ternary logic for balance ternary operation are given. These are ternary half adder, full adder, full subtract, comparator, multiplier, transmission gate, Multiplexer/Demultiplexer and arithmetic logic circuits.

SYNTHESIS, MINIMIZATION AND REALIZATION PROCEDURE FOR TERNARY SWITCHING FUNCTION

According to theorem 1 [93] any ternary function $F(x_1x_2 \dots x_n)$ may be generated from $x_1x_2 \dots x_n$ by canonical sum form is

$$F(x_1x_2 \dots x_n) = F_2(x_1x_2 \dots x_n) + 1 \cdot F_1(x_1x_2 \dots x_n)$$

Here F_2 is a function containing terms 2's or 0's & F_1 containing terms 1's & 0's. So, if function table is known for ternary operation, any function can be synthesized from function table.

Three basic methods to minimize ternary function are

- 1) Manipulation of algebra expression as in Boolean algebra.
- 2) The tabular method.

3) Ternary K. map method.

Out of these methods, ternary K.map method is adopted in this book for function minimization.

For the implementation of ternary circuits it is necessary to convert ternary variable in to unary variable. It is shown in Figure 5.2 and truth table in 5.1

Table 5.1: Truth table for converter.

Ternary variable input x	Ternary unary output		
	x^0	x^1	x^2
0	2	0	0
1	0	2	0
2	0	0	2

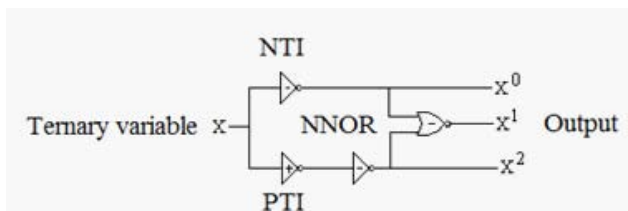


Figure 5.2: Ternary to Unary converter.

Ternary to unary decoder is a circuit build using ternary to unary converter and ternary to unary conversion is carried out using T-Inverters and OR gates.

HALF ADDER DESIGN

Ternary half adder is a circuit for the addition two 1 trit numbers is referred to as a half adder. Circuit does not consider a carry generated in the previous addition. Table 5.2 expresses the addition process in ternary logic system. Here A and B are two inputs and sum(S) and carry (C) are two outputs. Since no grouping of 2's and 1's is possible, the output equation is as below.

Table 5.2: Truth table for half adder.

INPUT		OUTPUT	
A	B	SUM S	CARRY C
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

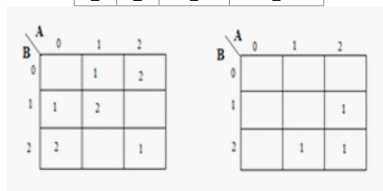


Figure 5.3: K-map for Half Adder.

Equation for Sum and Carry can be find from map as,

$$SUM = A^2B^0 + A^1B^1A^0B^2 + 1\bullet(A^1B^0 + A^0B^1 + A^2B^2) \text{ and } CARRY = 0 + 1\bullet(A^2B^1 + A^1B^2 + A^2B^2)$$

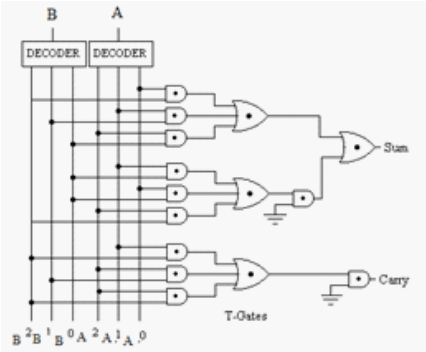


Figure 5.4: Half-Adder implementation using T-gates.

The grounding terminal of sun and carry shows logical grounding of logical 1 for balance system. Decoder block is ternary to unary decoder.

FULL ADDER DESIGN

A ternary full adder is a circuit that adds two inputs and previous carry generated. Truth table for full adder is given in Table 5.3 along with K-map in Figure 5.5, 5.6 shows full adder implementation.

Table 5.3: Truth table for full adder.

INPUT			OUTPUT	
A	B	C	SUM	CARRY
			S	C
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	2	2

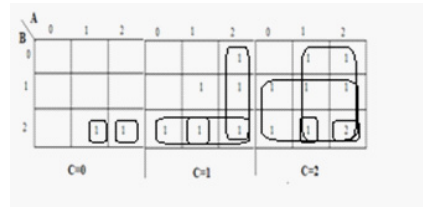
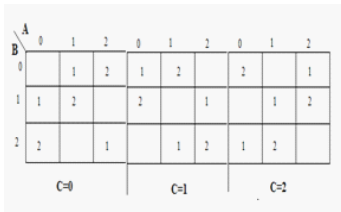


Figure 5.5: (a) Map for full adder SUM (b) Map for full adder CARRY with grouping.

K-map equations for sum and carry are to be,

$$\text{SUM} = A^2B^0C^0 + A^1B^0C^1 + A^0B^0C^2 + A^1B^1C^0 + A^0B^1C^1 + A^2B^1C^2 + A^0B^2C^0 + A^2B^2C^1 + A^1B^2C^2 + 1 \bullet (A^1B^0C^0 + A^0B^0C^1 + A^2B^0C^2 + A^0B^1C^0 + A^2B^1C^1 + A^1B^1C^2 + A^2B^2C^0 + A^1B^2C^1 + A^0B^2C^2)$$

$$\text{CARRY} = A^2B^2C^2 + 1 \bullet (A^2C^2 + A^0B^2 + A^2B^2 + A^1C^2 + A^2C^1 + B^2C^1 + B^1C^2 + B^2C^2 + A^1B^1C^1)$$

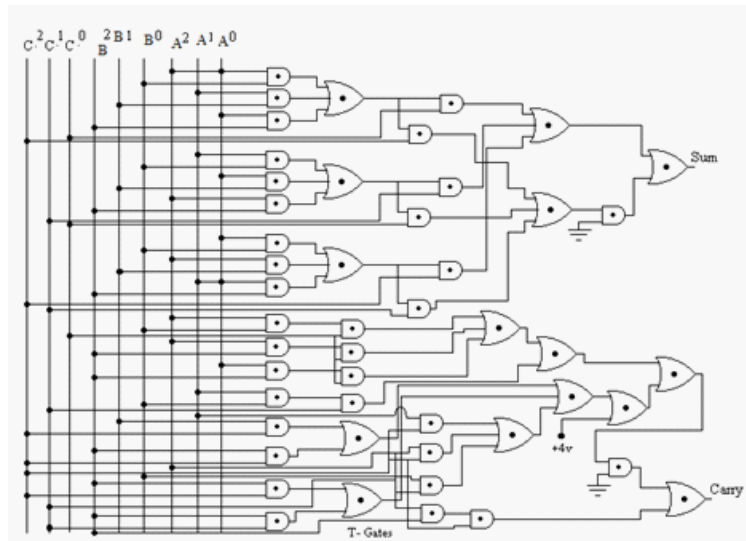


Figure 5.6: Full adder implementation.

FULL SUBTRACTOR

Ternary full-Subtractor is a circuit that subtracts two inputs & previous borrow. Truth table for Subtractor is given in Table 5.4 & k-map in Figure 5.7

Minimized equation for the Subtractor is

$$\text{Subtraction} = B_0^0 [X_1^0 X_2^1 + X_1^2 X_2^1 X_2^2] + B_0^1 [X_1^0 X_2^0 + X_1^1 X_2^1 + X_1^2 X_2^2] + B_0^2 [X_1^1 X_2^0 + X_1^0 X_2^2 + X_1^2 X_2^1] + 1 \bullet [B_0^0 [X_1^1 X_2^0 + X_1^2 X_2^1 + X_1^0 X_2^2] + B_0^1 [X_1^0 X_2^0 + X_1^1 X_2^2] + B_0^2 [X_1^0 X_2^0 + X_1^1 X_2^1 + X_1^2 X_2^2]]$$

$$\text{Borrow} = B_0^2 [2[X_2^1 + X_2^2]] + 2[X_1^0 X_2^1 + X_1^0 X_2^2] + 2 \cdot [X_1^1 X_2^2] + X_1^2 X_2^2 B_0^1 + X_1^1 X_2^0 B_0^2$$

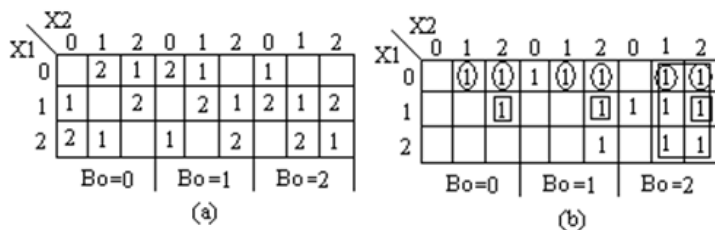


Figure 5.7: k-map for Full Subtractor.

X1	X2	Bin	Subtraction	Borrow
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	1
1	0	0	1	0
1	0	1	0	0
1	0	2	2	1
1	1	0	0	0
1	1	1	2	0
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 5.4: Truth table for Subtractor.

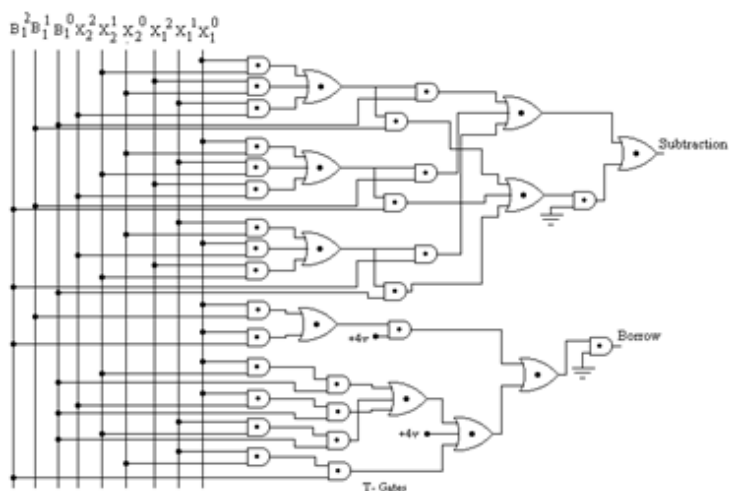


Figure 5.8: circuit implementation of Full-Subtractor.

TERNARY COMPARATORS

Ternary comparator circuit compares two input X_1 & X_2 & according generates output as $X_1 = X_2$, $X_1 > X_2$, $X_1 < X_2$. Truth table for comparator is show in table 5.8 along with Kmap in Figure 5.7.

Output equation for $X_1 = X_2$, $X_1 > X_2$, & $X_1 < X_2$ are

$$F_{X_1=X_2} = X_1^0 X_2^0 + X_1^1 X_2^1 + X_1^2 X_2^2$$

$$F_{X_1 > X_2} = X_1^1 X_2^0 + X_1^2 X_2^0 + X_1^2 X_2^1$$

$$F_{X_1 < X_2} = X_1^0 X_2^1 + X_1^0 X_2^2 + X_1^1 X_2^2$$

Table 5.5: Truth table for Comparator.

X_1	X_2	$X_1 = X_2$	$X_1 > X_2$	$X_1 < X_2$
0	0	2	0	0
0	1	0	0	2
0	2	0	0	2
1	0	0	2	0
1	1	2	0	0
1	2	0	0	2
2	0	0	2	0
2	1	0	2	0
2	2	0	0	2

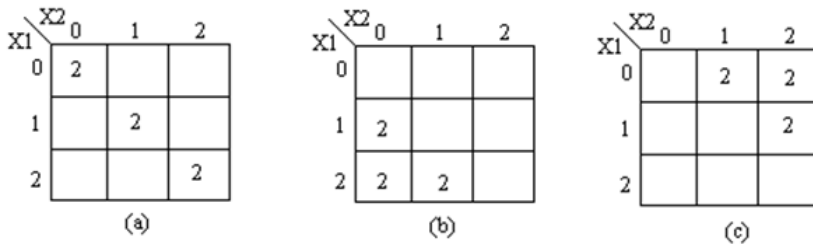


Figure 5.9: k-map for comparator (a) $X_1 = X_2$ (b) $x_1 > x_2$ (c) $x_1 < x_2$.

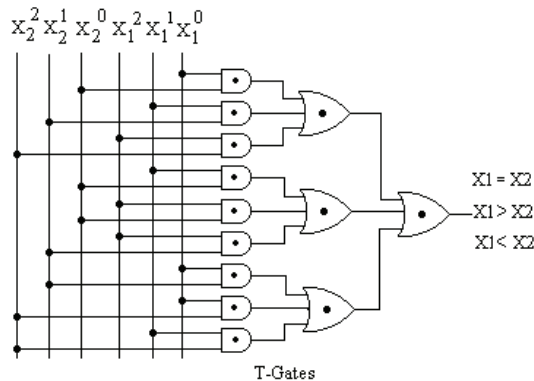


Figure 5.10: Show implementation of comparator.

TERNARY MULTIPLIER

Ternary multiplier is a circuit that multiplies two input numbers & generates corresponding product. Truth table for this circuit shows in Table 5.9 & corresponding K- map in Figure 5.9.

Table 5.6: Truth table for multiplier.

X1	X2	Product	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

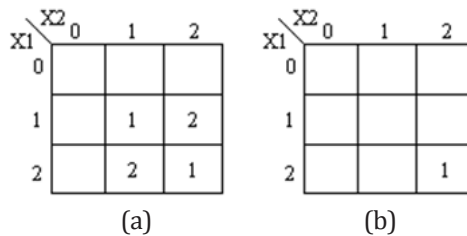


Figure 5.11: K-map for multiplier (a) Product (b) Carry.

From K-map output equations are:

$$F_{prod} = F = X_1^1 \cdot X_2^2 + X_2^1 \cdot X_1^2 + [X_1^1 \cdot X_1^2 + X_2^1 \cdot X_2^2]$$

$$F_{carry} = F = 1 \cdot X_1^2 \cdot X_2^2$$

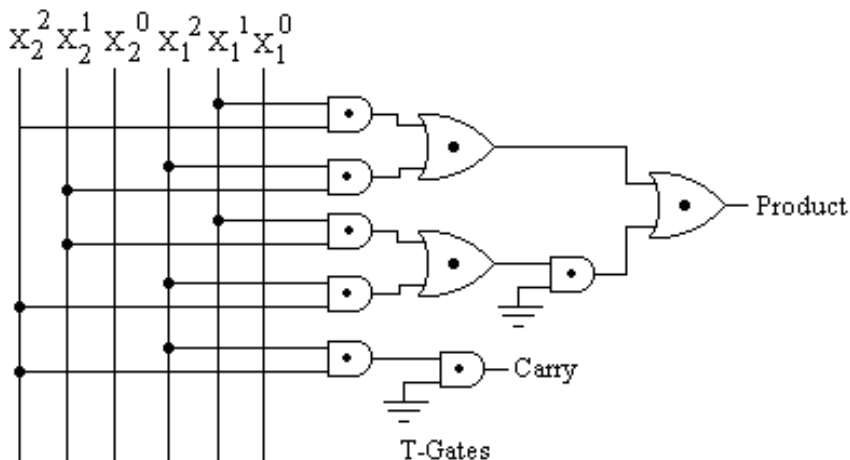


Figure 5.12: Implementation of multiplier.

TRANSMISSION GATE (TG)

CMOS TG is shown in Figure 5.11 along with its symbol. It is designed using P of N channel MOSFET. TG allows V_{in} to be connected to output when enable of keeps input & output isolated when disable. If $V_{G1} = -V_{D1}$ and $V_{G2} = +V_{DD}$, both transistors are on. The input voltage V_{in} is then connected to output through the parallel on resistance of the channels of the two transistors. As V_{in} approaches $+V_{DD}$, the N-channel device cuts off but the P-channel device remains nonsaturated. Therefore there is always a nonsaturated transistor between input and output. Moreover, no offset voltage exists when the output current is zero.

If $V_{G1} = V_{DD}$ and $V_{G2} = -V_{DD}$, both transistors are off, and the resistance between V_{in} and V_{out} is of the order of $10^9\Omega$

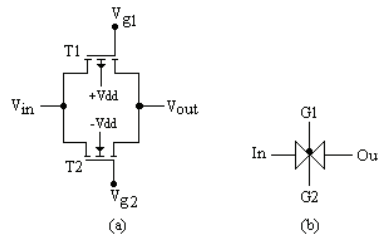


Figure 5.13: Implementation of CMOS TG & its symbol.

TERNARY MULTIPLEXER/DEMULTIPLEXER

Multiplexer is a circuit having multiple inputs and a single output. It is also known as decoder. The output function of the multiplexer is determined by the number of function lines. Thus for 2 trit multiplexer the output will be $3^2 = 9$ and two will be function select lines. Multiplexer i.e. function selection logic selects 1 out of 9 functions as an output. Function select logic is implemented using logic gates. The output equation of function selection logic is:

$$Z = X^0.Y^0 + X^0.Y^1 + X^0.Y^2 + X^1.Y^0 + X^1.Y^1 + X^1.Y^2 + X^2.Y^0 + X^2.Y^1 + X^2.Y^2$$

Block diagram for multiplexer is given in Figure 5.12 and gate level circuit implementation is give in Figure 5.13.

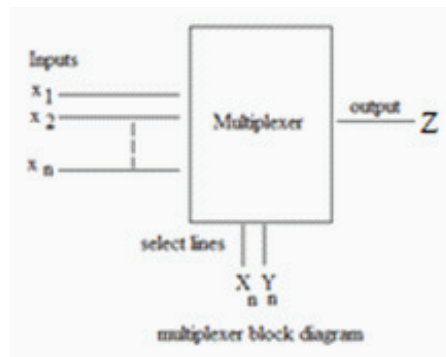


Figure 5.14: Block diagram of Multiplexer.

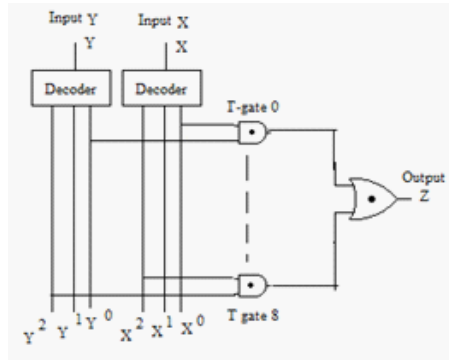


Figure 5.15: Gate level implementation of Multiplexer.

Table 5.7: Truth table for function selection logic.

Function select line		Output
X	Y	
0	0	E0
0	1	E1
0	2	E2
1	0	E3
1	1	E4
1	2	E5
2	0	E6
2	1	E7
2	2	E8

	Y 0	Y 1	Y 2
X 0	E0	E1	E2
X 1	E3	E4	E5
X 2	E6	E7	E8

Figure 5.16: k-map for Multiplexer.

Demultiplexer is also referred to as encoder. Its functionality is reverse to that of multiplexer. It accepts the single input and distributes it over several outputs [1]. Block diagram for Demultiplexer is shown in Figure 5.14. Gate level implementation is left as an exercise to reader.

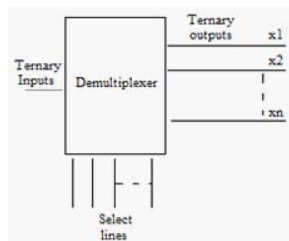


Figure 5.17: Ternary Demultiplexer.

TERNARY ARITHMETIC AND LOGIC UNIT (T-ALU)

Ternary Arithmetic Logic Unit (ALU) is a digital circuit used to perform arithmetic and logic operations [2,3]. It represents the fundamental building block of the Central Processing Unit (CPU) of a ternary computer. Architecture of 1 trit ALU is given in Figure: 5.14. It operates on $\pm 5v$ dual

power supply. The three logic levels of ternary circuit by states 0, 1, & 2 as

0 = -5V, 1 = Ground Potential and 2 = +5V

ALU carries out arithmetic operations like addition, subtraction, multiplication, and logic operations compare, NAND, NOR, NOT, AND, and OR.

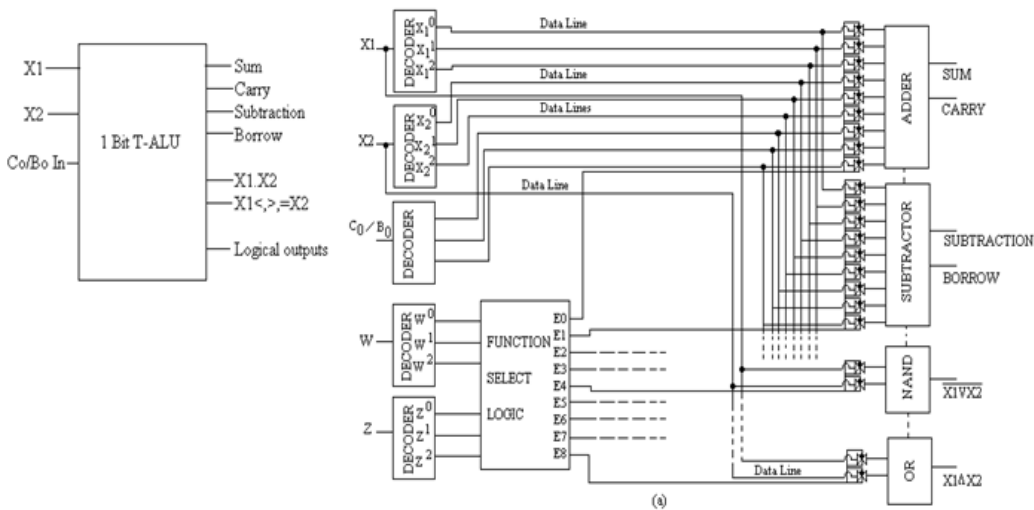


Figure 5.18: An architecture of 1 trit ALU.

Table 5.8: Table truth table for T-ALU.

Function select lines		Functions	Output equations
W	Z		
0	0	Addition	$X1 + X2$
0	1	Subtraction	$X1 - X2$
0	2	Multiplication	$X1 * X2$
1	0	Compare	$X1 >, <, = X2$
1	1	NAND	$\overline{X1 \wedge X2}$
1	2	NOR	$\overline{X1 \vee X2}$
2	0	NOT	$\overline{X1}$
2	1	AND	$X1 \wedge X2$
2	2	OR	$X1 \vee X2$

Basic building blocks of ALU are decoders, function select logic (Multiplexer), transmission gate and separate processing modules. Explanation about each module is given earlier in this chapter.

Function selection logic selects 1 out of 9 functions listed in Table 5.8, depending upon the logic state on function select lines W & Z. Output lines of selection logic are connected to TG associated with each module. Any module is selected only when associated TG is enabled else it is isolated from data lines. For e.g. If input from select lines W & Z = 0, output E_0 of selection logic is high (2) while E_1 to E_8 is low (0) so, TG associated with adder module will be enable allowing data lines to be connected to adder modules while other modules are isolated from the data lines. Gate level module implementation is given in the Figure 5.18 for few modules.

By cascading 'n/2' slices 'n' trit ALU can be formed. Architecture for 2 trit ALU is shown in Figure 5.19.

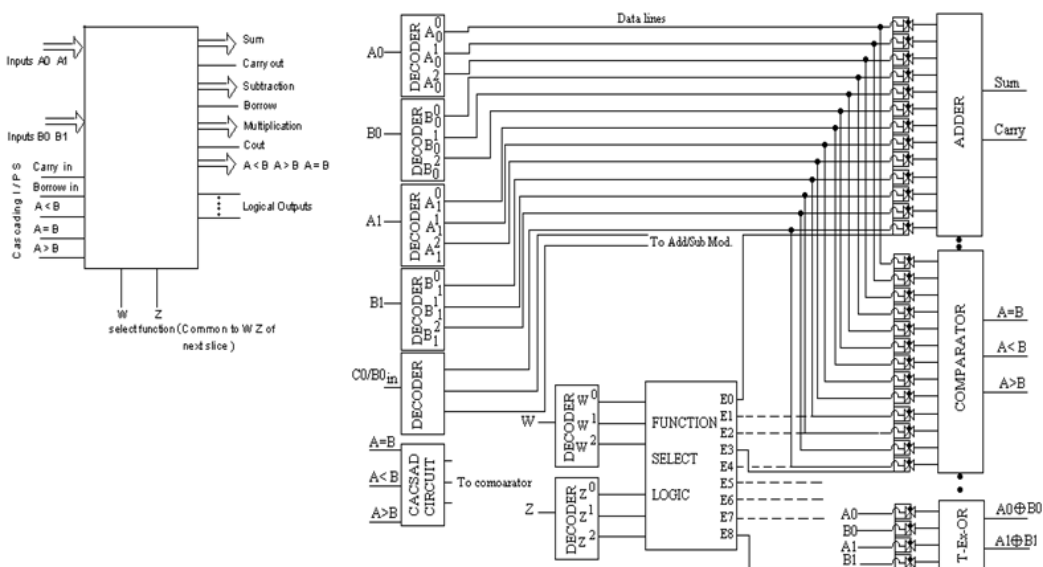


Figure 5.19: 2- trit ALU design.

For 2- trit ALU design cascading of 1- trit ALU is needed. Certainly there will be modifications while cascading two 1-trit modules. For an example, full adder, carry generated in the first slice is to be considering in next slice, same is the case with Subtractor, multiplier and comparator.

Design of 2-Trit Full Adder/Subtractor

For designing 2-trit adder, two full 1-trit adders are required. If slice is not connected in cascade, input C0/B0 should be connected to -5v. Block diagram for adder module is shown in Figure 5.20. (a) along with K-Map and truth table is given in Appendix II.

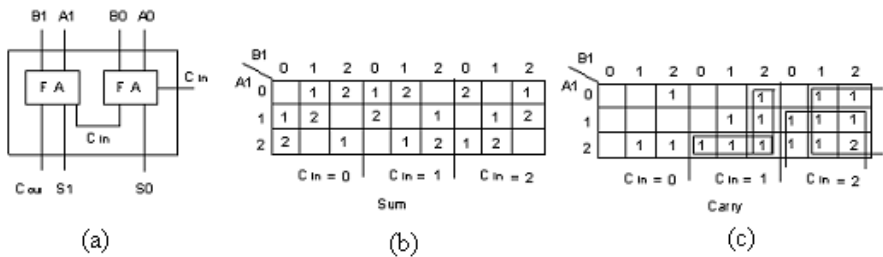


Figure 5.20: (a) 2-trit adder (b) k-map for sum (c) k-map for carry

From K-map output equation for full adder is:

$$F_{\text{sum}} = C_{in}^0 [A_1^2 B_1^0 + A_1^1 B_1^1 + A_1^0 B_1^2] + C_{in}^1 [A_1^1 B_0^1 + A_0^0 B_0^1 + A_1^2 B_1^2] + C_{in}^2 [A_1^0 B_1^0 + A_1^2 B_1^1 + A_1^1 B_1^2] + 1. [[C_{in}^0 (A_1^1 B_1^0 + A_1^0 B_1^1 + A_1^2 B_1^2)] + [C_{in}^1 (A_1^0 B_1^0 + A_1^2 B_1^1 + A_1^1 B_1^2)] + [C_{in}^2 (A_1^2 B_1^0 + A_1^1 B_1^1 + A_1^0 B_1^2)]]$$

$$F_{\text{carry}} = A_1^2 B_1^2 C_{in}^2 + 1. \{ [A_1^2 B_1^1 C_{in}^0 + A_1^2 B_1^2 C_{in}^0 + A_1^0 B_1^2 C_{in}^0 + A_1^1 B_1^1 C_{in}^1] + 2. [(B_1^1 C_{in}^2 + B_1^2 C_{in}^2) + (B_1^2 C_{in}^2 + B_1^2 C_{in}^1) + (A_1^1 C_{in}^2 + A_1^2 C_{in}^2) + (A_1^1 C_{in}^1)] \}$$

Circuit implementation for full adder is same as shown in Fig.5.6. On the same basis full Subtractor is constructed. Truth table for Subtractor is shown in appendix II.

Design of Multiplier Module

Multiplication of n-bit ternary number requires generation of partial product, shifting operations & finally addition of partial product. Implementation of multiplier blocks as a combination of 1-trit ternary multiplier, half and full T-adders. Table 6.7 (a) shows rules for multiplication, (b) truth table for 1-bit multiplier along with K-Map for 1-trit multiplier. Figure 5.17 Shows implemented ternary multiplier blocks. Truth table for the same is given in Appendix II from K-map, the output equation of multiplier is:

$$F_{\text{mul}} = A_0^2 B_0^1 + B_0^2 A_0^1 + 1. (A_0^1 B_0^1 + A_0^2 B_0^2)$$

$$F_{\text{carry}} = 1. (A_0^2 + B_0^2)$$

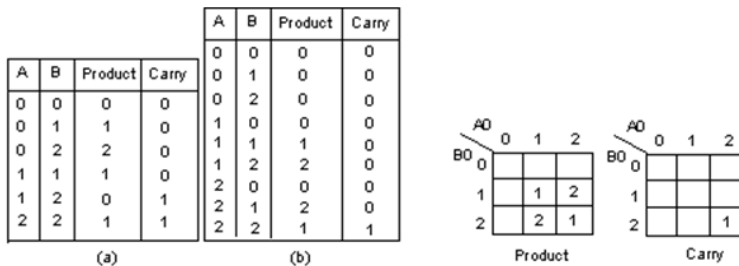


Figure 5.21: (a) shows rules for multiplication, (b) truth table for 1-bit multiplier and K-Map for 1-trit multiplier.

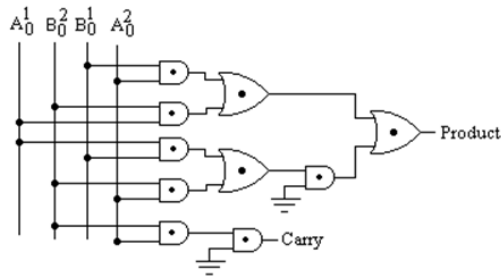


Figure 5.22: T- Gate

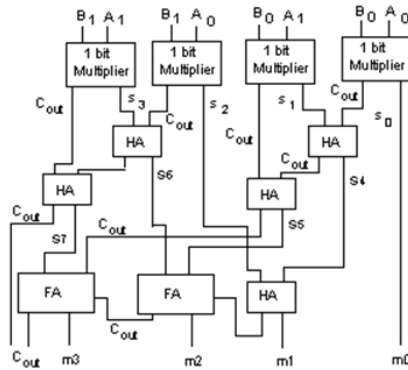
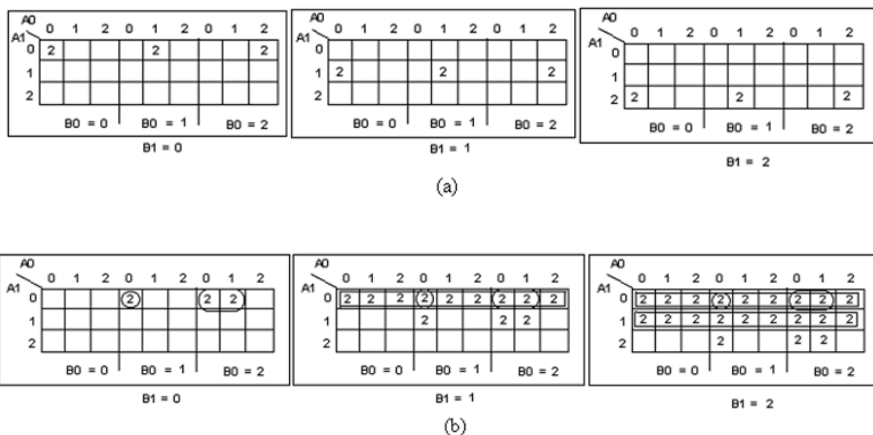


Figure 5.23: 2-bit Ternary multiplier

Design of Comparator Module

2-trit comparator is a circuit that compares 2-trit $A = A_0, A_1$ with $B = B_0, B_1$ and its output is either $A=B$, $A>B$ or $A<B$. Block diagram for comparator is shown in Figure 5.18 and its truth table is given in Appendix II. Figure 5.19 shows K-map for above three conditions.



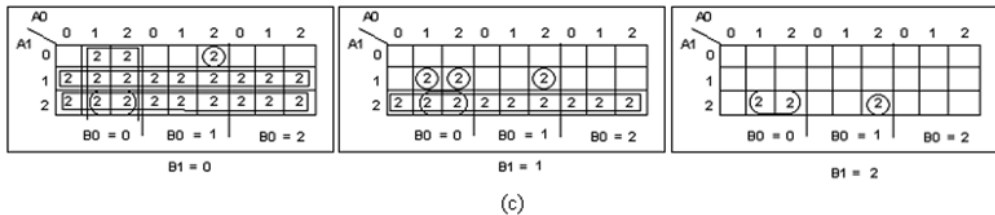


Figure 5.24: (a) K-map for A=B (b) K-map for A<B (c) K-map A>B.

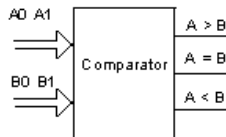


Figure 5.25: Block diagram for comparator.

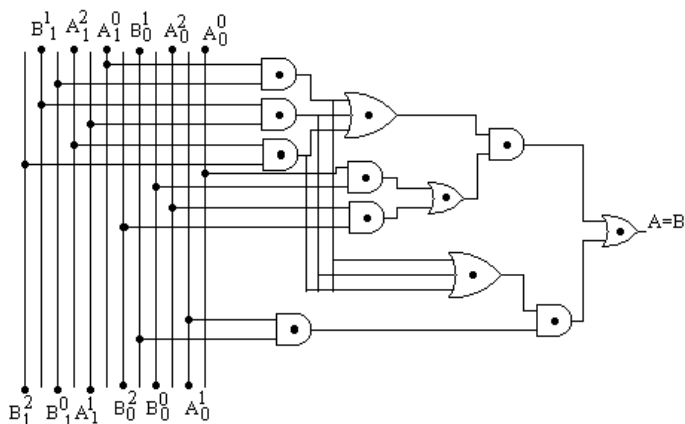


Figure 5.26: Circuit implementation comparator.

From k-map output equations are

$$A=B = (A_0^1 B_1^0 + A_1^1 B_1^1 + A_1^2 B_1^2) [A_0^0 B_0^0 + A_0^2 B_0^2] + A_0^1 B_0^1 [A_1^0 B_1^0 + A_1^1 B_1^1 + A_1^2 B_1^2]$$

$$A<B = A_0^0 A_1^1 B_0^1 B_1^1 + A_0^0 A_1^2 B_0^1 B_1^2 + A_0^1 A_1^1 B_0^2 B_1^1 + 2A_1^0 B_1^1 + 2A_1^0 B_1^2 + 2A_1^1 B_1^2 + 2B_0^2 B_1^2 [A_0^0 + A_0^1] + A_0^0 A_1^0 B_0^1 + 2A_1^0 B_0^2 [A_0^0 + A_0^1]$$

$$A>B = A_0^2 A_1^1 B_0^0 B_1^1 + A_0^1 A_1^1 B_0^0 B_1^1 + A_0^2 A_1^1 B_0^1 B_1^1 + A_0^2 A_1^2 B_0^1 B_1^1 + A_0^2 A_1^2 B_0^1 + 2B_0^0$$

$$B_1^0 [A_0^1 + A_0^2] + 2A_1^2 B_0^0 [A_0^1 + A_0^2] + 2A_1^1 B_1^0 + 2A_1^2 B_1^1 + 2A_1^2 B_1^0$$

Design of Ex-OR Module

Ternary Ex-OR function is mod-3 addition of ternary numbers & neglecting carry generated. Ex-OR function is implemented using half adders. Figure 7.8 is block for Ex-OR module. Rules for mod-3 addition is given in Table 7.6 (a) for 1-trit Ex-OR & truth table for Ex-Oring of 2-bit is in Appendix II. Ex-OR gates discussed above can directly use for circuit implementation.

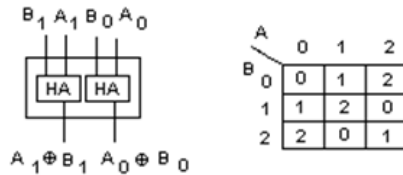


Figure 5.27: (a) Ex-OR implementation (b) Rules for 1-trit Ex-OR.

Design of Logical Operation Modules

For 2-bit logical operation T-gates are used Figure 7.9 shows block diagram for logical AND/ NAND/OR/NOR operations.

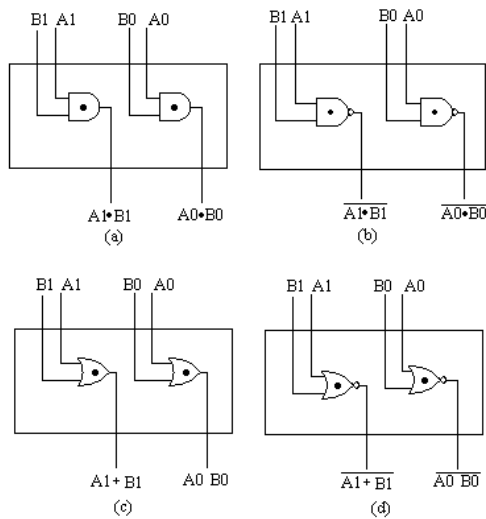


Figure 5.28: Block diagram for logical AND/NAND/OR/NOR.

Truth table for above gate operations is given in chapter 1.

References

1. Yoeli M, Rosenfeld G. Logical Design of ternary switching circuits. IEEE Trans. Computer. 1965; 14: 19-29.
2. Dhande AP, Ingole VT. Synthesis and Implementation of 1-Bit T-ALU. IEEE ACE, PUNE, India. 2003.
3. Dhande AP, Ingole VT. Design and Implementation Of 2 Bit Ternary ALU Slice. SETIT 2005, 3rd International Conference: Sciences Of Electronic, Technologies Of Information And Telecommunications, Tunisia. 2005; 7-21.