SETIT 2005
3RD INTERNATIONAL CONFERENCE: SCIENCES OF ELECTRONIC, TECHNOLOGIES OF INFORMATION AND TELECOMMUNICATIONS
MARCH 17-21, 2005 – TUNISIA

Design And Implementation Of 2 Bit Ternary ALU Slice

*A. P. Dhande **V. T. Ingole
* *Pune Institute Of Computer Technology, Pune, India* ***College Of Engineering, Badnera, India*
ashwindhande@pictsctr.edu, vijayingole@hotmail.com

Abstract

This paper describes the architecture, design & implementation of 2 bit ternary ALU (T-ALU) slice. The proposed ALU is designed for two-bit operation & can be used for n bit operations by cascading n/2 ALU slices. This ALU is implemented using C-MOS ternary logic gates (T-Gates) for ternary arithmetic & logic circuits. Ternary gates are implemented using enhancement / depletion MOSFET technology, thus proposed ALU is suitable for LSI / VLSI implementation. The designed technique used here requires only two stages i.e. decoder & T-gates, as against three stages i.e. decoder, binary gates & encoder require in conventional ternary logic implementation.

Index Terms *: Ternary, Unary function, T-Gates, Literal.*

I. Introduction

Alexander [1964] showed that natural base (e = 2.71828) is the most efficient radix for implementation of switching circuits. It seems that most efficient radix for the implementation of digital system is 3 than 2. Ternary logic system, meaning that it has 3 valued switching. Ternary system has several important advantages over binary. It can be summarized as reductions in the interconnections require to implement logic functions, thereby reducing chip area, more information can be transmitted over a given set of lines, lesser memory requirement for a given data length. Besides this serial & some serial-parallel operations can be carried out at higher speed [1][2][3]. Its advantages have been confirmed in the application like memories, communications and digital signal processing etc. [7].

It has been proven that realization & implementation of combinational & sequential function is possible for ternary systems [4][5][6][7]. The implementation is based around bipolar transistors, MOSFETs etc. a basic switching elements, which is refereed to as T-Gates [8]. Besides this several authors have proposed reduction techniques to realize ternary functions [9][10][11][12].

In this contribution, we propose ALU capable of performing basic ternary arithmetic & logic operations as mentioned in table 1. We also suggest a scheme that takes the advantage of minimization techniques proposed by [9][11][13] & implemented using T-gates designed for ternary operations. This scheme shows reduction in the number of gate count to implement ternary functions. Firstly we describe the design of 2 bit ALU and then integrate over ALU slice.

The organization of paper is: Section II describes basic T-Gate implementation, 2 bit ALU architecture is given in section III, section IV describes 2 bit ALU design and ALU slice design. Experimental results & performance evaluation is given in section V. Finally conclusion is given in section VI.

Table 1:Functional Table of T-ALU

Function select lines W	Z	Functions
0	0	Addition
0	1	Subtraction
0	2	Multiplication
1	0	Compare
1	1	OR
1	2	NOR
2	0	AND
2	1	NAND
2	2	Ex-OR

II. Ternary logic Gates & Their Implementation

The algebra proposed by Rosenfeld – yoeli [9], which is suitable for arithmetic operations [14] can be realized through ternary logic gates. In this paper three logic levels of T- Gates are represented by states 0,1,2 as –Vcc (-5v), zero potential (0v) & +Vcc (+5v) respectively. The basic ternary inverters namely simple ternary inverter (STI), positive ternary inverter (PTI) & negative ternary inverter (NTI) forms an operator set that is complete in logic sense. All these inverters are combined to realize ternary functions like NAND (T-NAND), NOR (T-NOR) T-AND, T-OR, T-EX-OR etc.

In general ternary inverter, AND & OR functions are defined as:

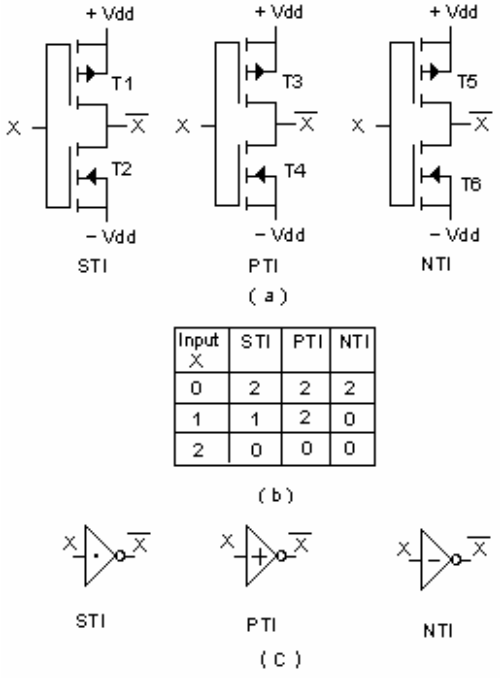
STI = $\overline{x}^i = 2 - x$

PTI & NTI = $\overline{x}^i = i$ if $x = i$

$= 2 - i$ if $x \neq i$ where i can be 2 or 0

TOR = Max (x, y) & TAND = Min (x, y)

Fig.1 (a) shows implementation of basic ternary inverters, (b) truth table & (c) symbols for inverters respectively. Table 2 gives threshold values v_T requires to implement ternary inverters. Depending upon inverters used, positive ternary NAND (PTNAND), negative ternary AND (NTAND) etc. functions can realize. We have implemented T-Gates using P & N channel enhancement / depletion MOSFETs. Figure 2 (a) shows T-NAND, T-NOR implementation, (b) symbols & (c) truth table.



Input X	STI	PTI	NTI
0	2	2	2
1	1	2	0
2	0	0	0

Fig.1 (a) Implementation of T-Gates.

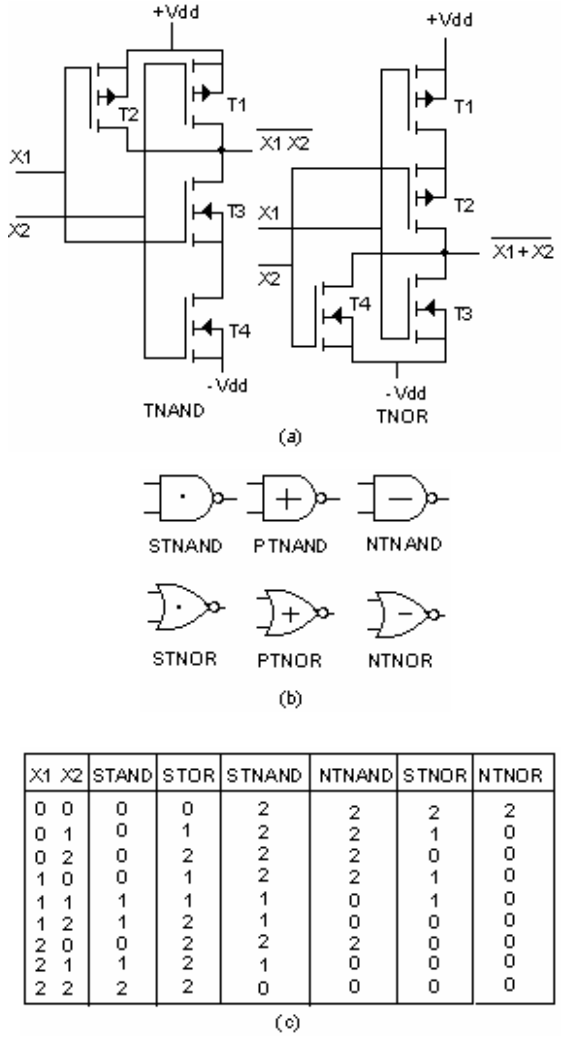
(b) Truth table for ternary inverters.

(c) Symbols for T-Gates.

For the implementation of standard ternary inverters 1MΩ resistances are actually connected between the drains of MOSFETS.

Table 2. Threshold values for inverters

Inverters	Transistor	Device type	V_{th}	Device	Drain ohmic resistance
STI	T1	p channel-enhancement	- 0.45v	international IRFD 9110	1MΩ
	T2	n channel-enhancement	0 v	international IRF 360	1MΩ
PTI	T3	p channel-enhancement	- 0.25	International IRF 9640	0.44143 Ω
	T4	n channel-enhancement	4.137	Harris IRF222	0. 67 Ω
NTI	T5	p channel-enhancement	2.69 v	Harris IRF 220	0.36731Ω
	T6	n channel-enhancement	-5v	Fairchild FQB9p25	0 .62Ω



X1	X2	STAND	STOR	STNAND	NTNAND	STNOR	NTNOR
0	0	0	0	2	2	2	2
0	1	0	1	2	2	1	0
0	2	0	2	2	2	0	0
1	0	0	1	2	2	1	0
1	1	1	1	1	0	1	0
1	2	1	2	1	0	0	0
2	0	0	2	2	2	0	0
2	1	1	2	1	0	0	0
2	2	2	2	0	0	0	0

(c)

Fig.2. (a) T-NAND T-NOR Implementation
(b) Symbols (c) Truth table

III. T-ALU Architecture & Working

Block diagram & architecture of proposed T-ALU is shown in fig.3 (a) and 3(b). Its functions are given in table 1.The operating voltages required are +5 & -5v.Main building blocks of ALU are function selection logic, C-MOS transmission gates & separate processing modules like adder, subtractor, comparator etc.

Function selection lines W & Z activates respective module by enabling C-MOS transmission gate (TG), while other modules are deactivated from data lines (decoder output lines). Output equation of function block is $Y = w^0 z^0 + w^0 z^1 + w^0 z^2 + w^1 z^0 + w^1 z^1 + w^1 z^2 + w^2 z^0 + w^2 z^1 + w^2 z^2$. When function selection lines are activated by ternary I/P, O/P of selection logic is high i.e.2 for corresponding I/P. This high O/P enable TG associated with respective module which connects data lines to the module while other modules are isolated from data lines. When TG [15] enable signal is low i.e.0, TG is opened (high impedance state =1v). Thus any function listed in table 1 can be realized.

Decoder in the circuit generates unary functions for input variable x as $x^0, x^1 \& x^2$ which is used for ternary function implementation. Table 4 is unary function table for I/P x. Fig.4 gives implementation of decoder by T-Gates.In fig.5 (a) implementation of TG & (b) symbol is given. Fig.6 describe implementation of ternary function with the designed technique used in this paper & Fig.7 shows connectivity of single module with TG & decoder.

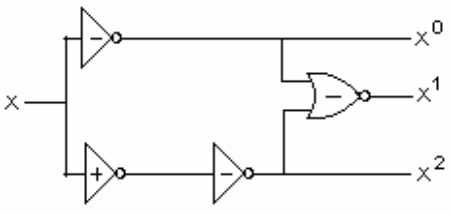


Fig.4. Implementation of decoder

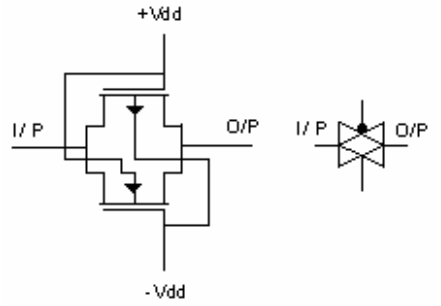


Fig.5. (a) Implementation of TG
(b) Symbol for TG

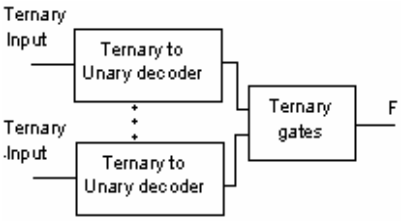
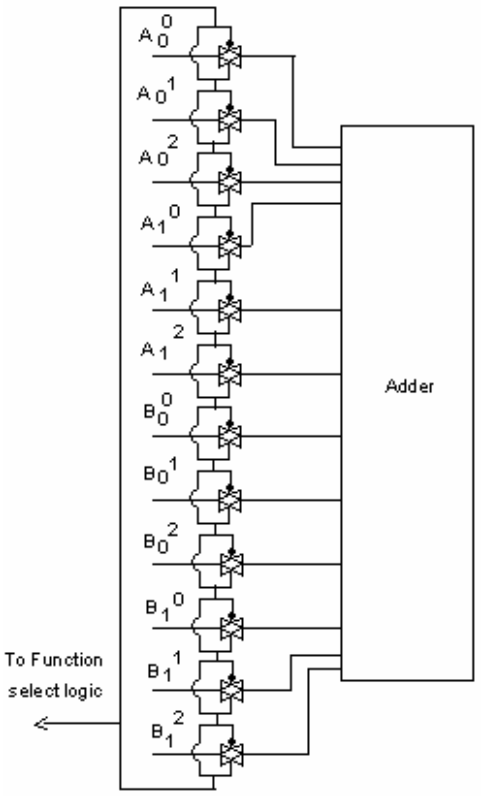


Fig.6. Implementation of ternary function.

Table.4 Unary function table

X	X^0	X^1	X^2
0	2	0	0
1	0	2	0
2	0	0	2



& selection logic

Fig.7. Connectivity of TG with module

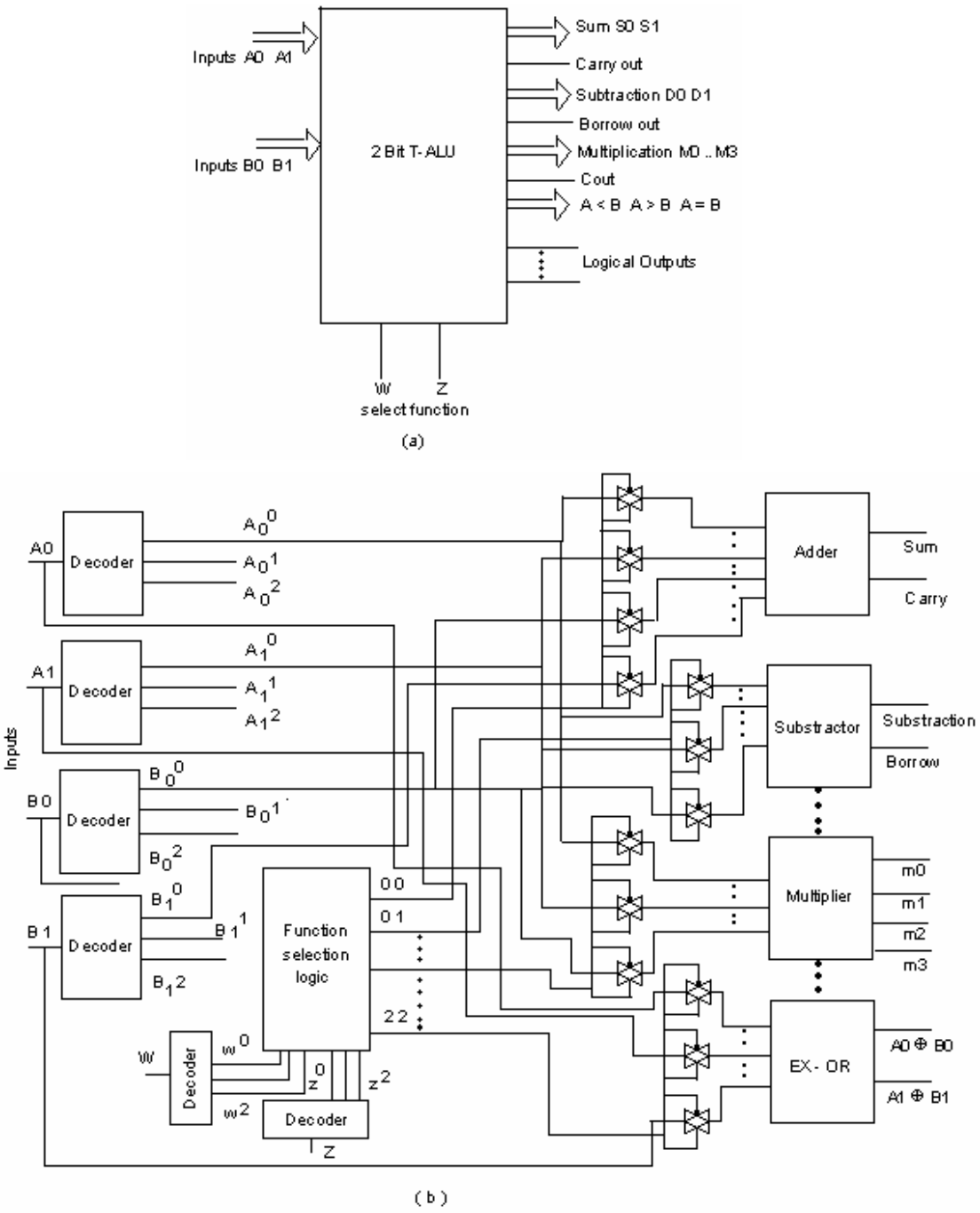


Fig.3. (a) Block diagram of T-ALU
(b) Architecture of T-ALU

IV. T-ALU Design

Design of ALU is based on ternary K- map method for ternary function minimization [9], out of various methods suggested in ref. [10] [11] [12] [13]. Here we describe the design of adder, multiplier Ex-OR & comparator modules. Same design concept is implemented for designing other modules.

IV (A). Design of Adder module

Design rules for ternary addition is given in the table-5. For two-bit addition of ternary numbers one full & one half-ternary adder is required. Truth table for half & full adder is given in appendix**I**.Fig.8 (a) shows block diagram of adder module (b) ternary k-map for half adder & (c) ternary k-map for full adder. Fig.9 shows T-gate implementation of half adder. The output equations for half adder are Sum $= A_0^2\ B_0^0 + A_0^1\ B_0^1 + A_0^0\ B_0^2 + 1.$ $(A_0^1\ B_0^0 + A_0^0\ B_0^1 + A_0^2\ B_0^2)$

Carry = 1. $(A_0^2\ B_0^1 + A_0^2\ B_0^2 + A_0^1\ B_0^2)$ Similarly, output equations for full adder is

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

Carry = $A_1^2\ B_1^2\ C_{in}^2 + 1.[\,A_1^2\ B_1^1\ C_{in}^0 + A_1^2\ B_1^2\ C_{in}^0 + A_1^0\ B_1^2\ C_{in}^0 + A_1^1\ B_1^1\ C_{in}^1\,] + 2\,[(B_1^1\ C_{in}^2 + B_1^2\ C_{in}^2) + (B_1^2\ C_{in}^2 + B_1^2\ C_{in}^1) + (A_1^1\ C_{in}^2 + A_1^2\ C_{in}^2) + (A_1^2\ C_{in}^1)]$

Table 5. Design rules for ternary addition

A	B	Sum	Carry
0	0	0	0
0	1	1	0
0	2	2	0
1	1	2	0
1	2	0	1
2	2	1	1

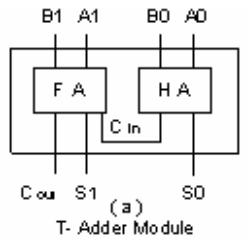
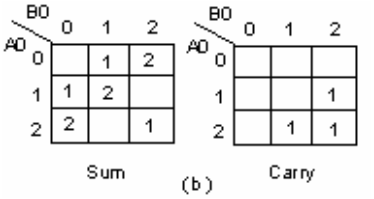


Fig.8 (a) Adder module,
(b) K-map for half adder
(c) K-map for full adder



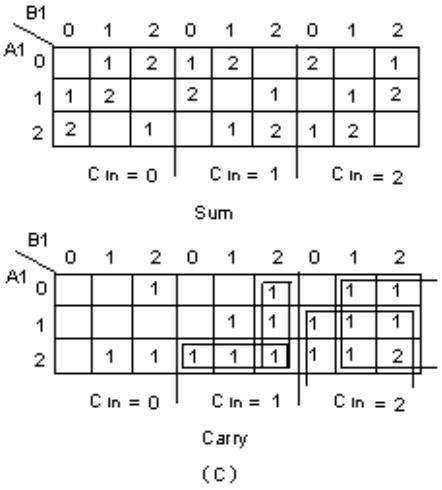
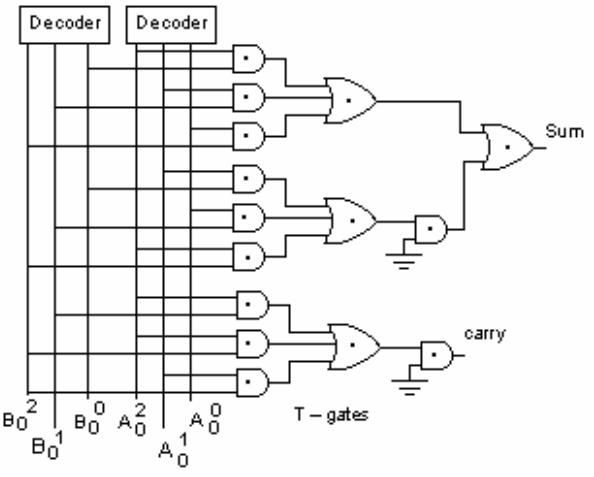


Fig.9. Implementation of half adder



IV (B). Design of Multiplier module

Multiplication of n-bit ternary number requires generation of partial product, shifting operations & finally addition of partial product. We have implemented multiplier block as a combination of 1-bit ternary multiplier, half and full T-adders.Table 6 (a) shows rules for multiplication,(b) truth table for 1-bit multiplier along with K-Map for 1-bit multiplier. Fig.10 Shows implemented ternary multiplier blocks. The output equation of 1 bit multiplier is Fmul = $A_0^2\ B_0^1 + B_0^2\ A_0^1 + 1.\ (A_0^1\ B_0^1 + A_0^2\ B_0^2)$.

Fcarry = $1.\ (A_0^2 + B_0^2)$

Table-6 (a) Rules for multiplication
(b) Truth table for 1-bit multiplication

A	B	Product	Carry
0	0	0	0
0	1	1	0
0	2	2	0
1	1	1	0
1	2	0	1
2	2	1	1

(a)

A	B	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

(b)

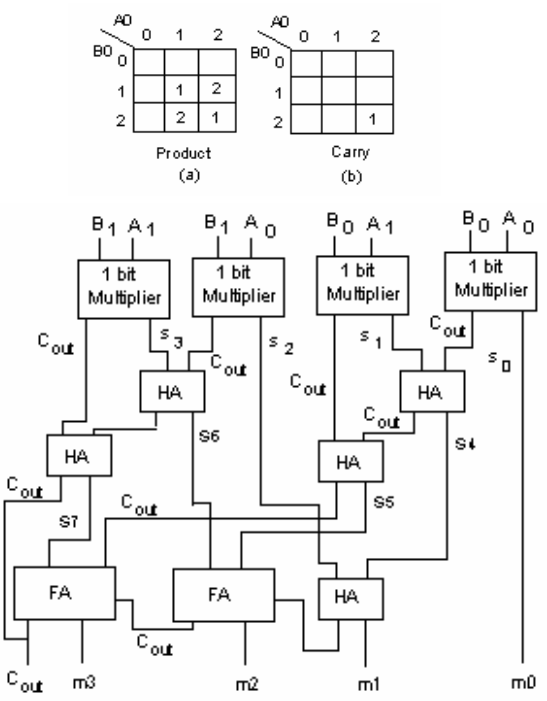


Fig.10 1bit ternary multiplier

IV (C). Design of comparator module

Block diagram for 2bit comparator is shown in fig.11. Appendix II gives truth table for comparator. Ternary k-map for condition A=B, A>B & A<B where A= A0,A1 & B= B0,B1is shown below. From k-map output equations are

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

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

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

$A_0^1\ A_1^1\ B_0^2\ B_1^1 + 2A_1^0\ B_1^1 + 2A_1^0\ B_1^2 + 2A_1^1\ B_1^2 +$

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

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

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

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

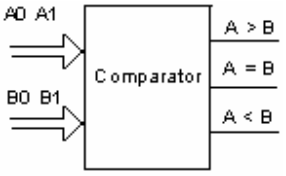
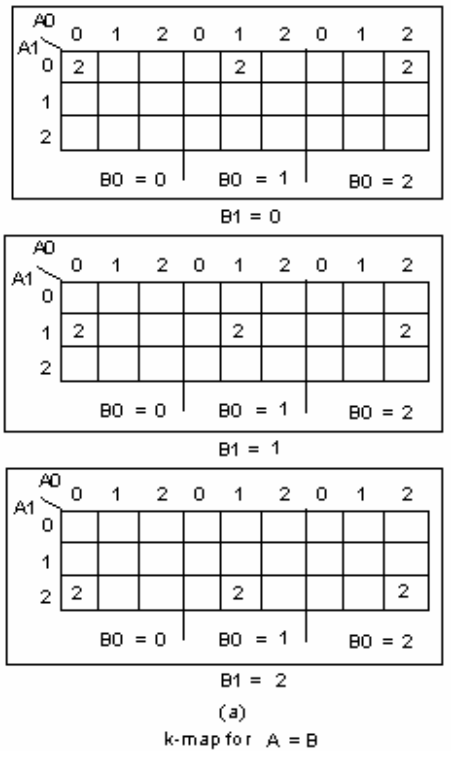
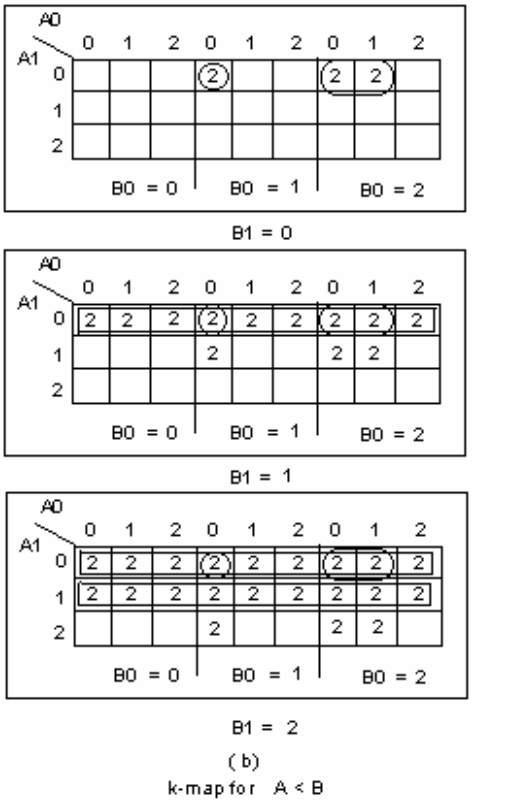


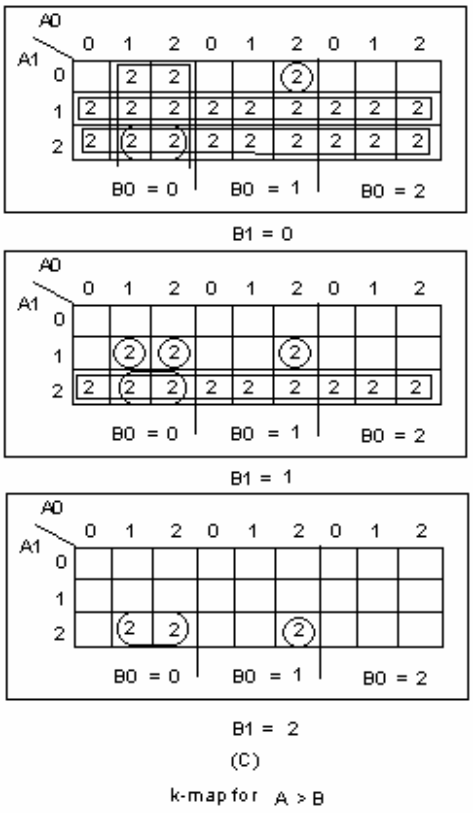
Fig.11.Comparator block



(a)
k-map for A = B



(b)
k-map for A < B



(C)
k-map for A > B

IV (d). Design of Ex-OR module

Ternary Ex-OR function is mod-3 addition of ternary numbers & neglecting carry generated [16]. Ex-OR function is implemented using half adders. Fig.12 is block for Ex-OR module. Rules for mod–3 addition is given in table 7 (a) & (b) is truth table for Ex-Oring of 2-bit number.

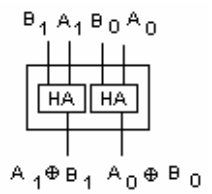


Fig. 12 Block for Ex-OR Module

Table 7 (a) Rules for Ex-OR operation
(b) Truth table for 2-bit Ex-OR

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

(a)

B_1	B_0	A_1	A_0	$A_0 \oplus B_0$	$A_1 \oplus B_1$
0	0	0	0	0	0
0	0	0	1	1	0
⋮	⋮	⋮	⋮	⋮	⋮
2	2	2	2	1	1

(b)

IV B. Implementation Of 2-Bit slice

The proposed design of 2-bit ALU is extended for implementing ALU slice. Block diagram of slice is shown in fig-13. Cascading inputs carry, borrow has to be connected carry, borrow, A=B,A>B,A<B to of next slice for n-bit slice else connect carry, borrow to ground & A=B,A>B,A<B to Vcc. Modifications required for ALU slice in 2bit ALU is shown in fig.14(a) (b)& (c). For Comparator, the output of present stage is connected to terminal 1, and previous stage output is connected to terminal 2 of circuit as in fig (c). For implementing n-bit logic functions no modification is required. For example to implement 4 bit logical AND operation, 2 slices are required out of which A0 A1 B0 B1 is to be connected slice 1 & A2 A3 B2 B3 is to be connected slice 2.

As the number of bits increases, multiplication becomes more complex. We have used the multiplier block as described in ref [3].

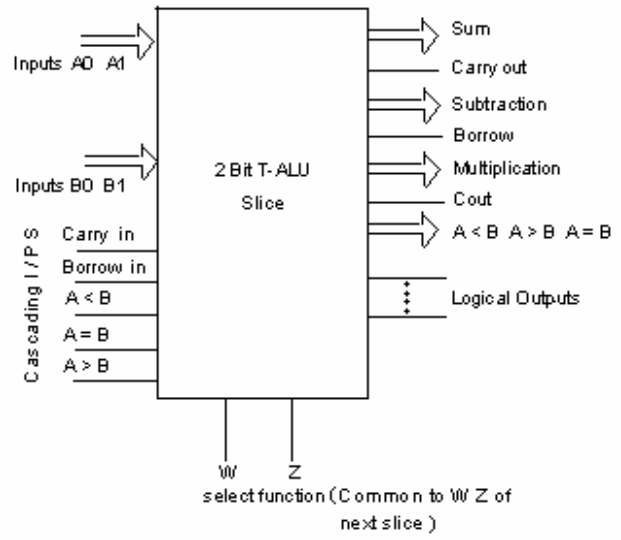


Fig.13 T-ALU Slice

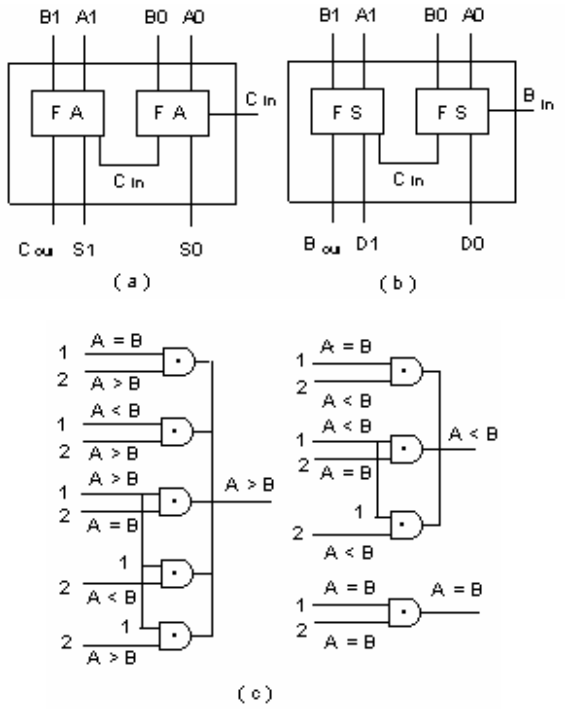


Fig.14 Modifications for ALU slice

V. Experimental verification & performance evolution

Experimental circuits are designed & implemented to verify workings of some modules of proposed ALU.These circuits are build using logic gates specified in section II. Typical experimental results are presented for adder, multiplier, & Ex-OR operations & as seen desired operations are achieved. Fig15(a) shows storage oscilloscope waveform for adder module when B1=1 B0= 2,A1=0, A0=2. , (b) is output of multiplier module when B1=2, B0=2, A1=2, A0=2 & finally (C) shows Ex-OR operation B1= 1, B0=0 ,A1=1, A0=0.

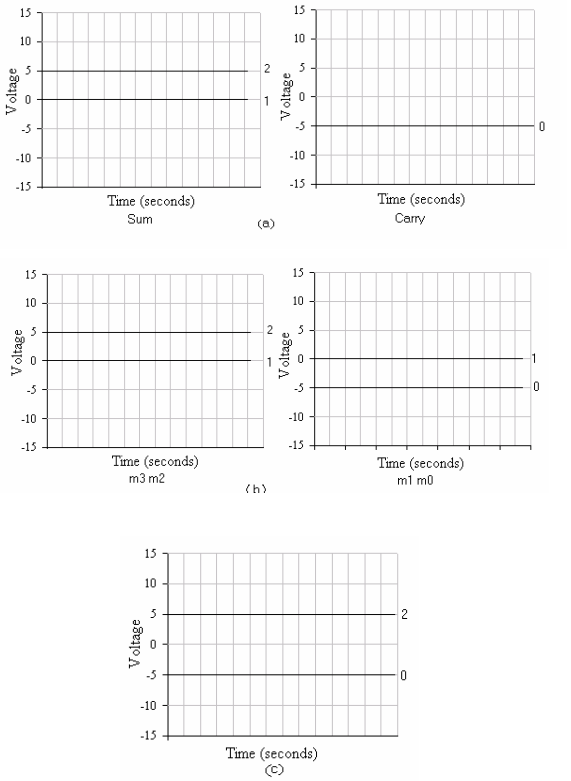


Fig.15 (a) DSO waveforms for Adder module.
(b) Multiplier output
(C) Ex-OR output

Performance evaluation

We make the perfomance evolution on the basis of implementation of ternary logic function in comparison with scheme proposed in [3] [5] [10] [13] with respect to hardware complexity. Here we compare number of gates

required for the implementation of half / full adders / substractor. A half adder of type [3][5][10][13] requires 13,20,17,30 binary gates respectively & encoder as an additional block for binary to ternary encoding where as in proposed scheme 15 T-gates are required without encoder. Same analogy can be extended for full adder. Here with the proposed scheme number go T-gates required are 56 compared to 108,115,83 & 120 binary gates for references [3][5][10][13].

Furthermore our T-gates have significant reduction in power dissipation, deduced propagation delay etc as compared with [3][5][10]. Table 8 gives average propagation delay comparison with T-gates proposed in [3]. These results are obtained by simulating the circuits on P-Spice simulation program.

Table 8:Propagation-delay comparison

	scheme based on T- Gates	scheme based on binary gates *
	Ave. propagation delay	Ave. propagation delay
Half adder	6 τ	7 τ
Full adder	7 τ	8 τ
Full substractor	6 τ	—

* Ref [3]
τ= Ave.propagation delay of 1 MOSFET

VI. Conclusion

A complete architecture, design & implementation of 2-bit ALU slice is describe. The scheme based on improved ternary logic gate is also described. It exhibits significant advantages of reduction in circuit complexity, low static power consumption & increased speed of operation.

The ALU can be extended to implement more function with modification in selection logic, decoder etc. Thus it can be incorporated as processing unit for ternary microprocessors.

The ternary logic functions can also be implemented using ternary PLAs [17] thereby increasing packing density, more complex function implementation etc. for future advancement.

VIII. References

1) D.I.porat "Three valued digital system" Proc.IEE Vol.116, No6, P.947-955, June 1969.
2) K.C.Smith "The prospects of multivalued logic technology & application view " IEEE transaction on computer, Vol.-C -30, P-619-627 September 1981.
3) P.C.Balla & A.Antoniou "low power dissipation MOS ternary logic family" IEEE journal on solid state circuits Vol. Sc-19 no-5, P.739-749, October 1984.
4) H.T.Mouhtaf & I.B.Jordan " Study on the implementation of three valued logic". Proc.ISMVL, P.359-372 May 1975.
5) Seiichi Muta "Micropower CMOS implementation of three valued logic function". Proc.ISMVL, P 61-63, May 1983.
6) Michel Israel & D.Etiemble " Some new results for ternary circuits"". Proc.ISMVL, P 66-69, May 1979.
7) Chung-Yu-Wu"Design & application of pipelined dynamic CMOS ternary logic & simple ternary differential logic" IEEE journal on solid state circuits Vol.28, No-8, August 1993.
8) T.Higuchi & M.Kameyama "Synthesis of optimal T-gate networks in multiple valued logic" Proc.ISMVL, P.190-195, May 1979.
9) G.Rosenfeld & M.Yeoli "Logical on electronic computers.P.19-29, V0l.EC-14 February 1965.
10) E.Mccluskey"Logic design of MOS ternary logic"Proc.ISMVL,P.1-5,May 1980.
11) T.C.Yang"Aminimization algorithm for ternary switching functions"Proc. ISMVL P.241-253, May 1975.
12) J.Huertas & J.Carmona "Low power ternary CMOS circuits.Proc.ISMVL P.170-174, May 1979.
13) T.Sasao “Compact SOP representation for multiple output functions-An encoding method using multiple valued logic” Proc.ISMVL 2001.
14) I.Halpern & M.Yoeli “ Ternary arithmetic unit” Proc.IEE, Vol.115,No.10,pp. 1385-1387, Oct. 1968
15) D.Hamilton & W.Howard "Basic Integrated Circuits Engineering"McGraw Hill, Motorola series in solid state electronics.
16) R.P.Hallworth & F.G.Heath “Semiconductor circuits for ternary logic” The institution of electrical engineers, Monograph No.482E Nov.1961, p.p.219-223.
17) T.Hanyu, M.kemayama,C.Zukeran “Multiple-valued mask programmable logic array using one-transistor universal literal circuits. Proc.ISMVL 2001.

Appendix-I

Truth-Table For Half & Full Adders

Half Adder

X2	X1	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

Full Adder

X2	X1	C in	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	1	1
2	1	1	2	1
2	1	2	0	1
2	2	0	0	1
2	2	1	1	1
2	2	2	2	2

The design procedure is

1) Construct truth table & k-map for input variable.
2) Find the cell grouping if possible. i.e. grouping of 1 x 3,2 x 3,3 x 3 of 2's term and 1's terms if possible. For grouping of 1's, 2 can be considered as don't care terms.
3) Deduce, minimize & realize the switching function.
4) The minimization equation is F = f1 + 1.f2

Appendix II

Truth tables for comparators & Multiplier

Multiplier Truth Table

B1	B0	A1	A0	m_3	m_2	m_1	m_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	2	0	0	0	0
0	1	1	0	0	0	1	0
0	1	2	1	0	0	2	1
0	1	2	2	0	0	2	2
0	2	0	0	0	0	0	0
0	2	1	1	0	0	2	2
0	2	2	2	0	1	2	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	2	0	1	2	0
1	1	1	0	0	1	1	0
1	1	2	1	1	0	0	1
1	1	2	2	1	0	1	2
1	2	0	0	0	0	0	0
1	2	1	1	0	2	0	2
1	2	2	2	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	0	0	0	0	0	0	0
2	0	0	1	0	0	2	0
2	0	1	2	1	0	1	0
2	1	1	0	0	2	1	0
2	1	2	1	1	2	1	1
2	1	2	2	2	0	0	2
2	2	0	0	0	0	0	0
2	2	1	2	0	1	1	2
2	2	2	1	2	1	0	1

Comparator Truth Table

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