ASIC Implementation of Low Power Area Efficient Folded Binary Comparator

N.Saravanakumar[#1], A. NirmalKumar[*2], A.Nandhakumar[#3], G.E.KanyaKumari[#4]

[#] Department of Electrical and Electronics Engineering,
Bannari Amman Institute of Technology, Sathyamangalam, Tamilnadu, India
[1] sarapalani81@gmail.com
[3] nandhakumarme@gmail.com
[4] kanyakumarige@bitsathy.ac.in
[*] Karpagam College of Engineering
Coimbatore, Tamilnadu, India
[2] ankhod@gmail.com

***Abstract*—ASIC implementation of a parallel binary comparator based on radix-2 tree structure, utilizing Carry Look Ahead (CLA) technique is proposed in this brief. This novel comparator architecture achieves both low power and high-speed operation, particularly at low-input data activity environments. The proposed comparator is designed using VHDL code and synthesized using ALTERA QUARTUS - II. Experimental evaluation of the proposed and state of-the-art designs revealed that the proposed comparator design exhibits a reduction in delay by 49.8% and gate count by 42.6% for a 16 bit design, compared to the best of the schemes used for comparison.**

Keyword - Binary comparator Digital Arithmetic, Tree Structure, Carry Look Ahead, Priority encoding.

I. Introduction

A digital comparator or magnitude comparator is a hardware electronic device that has two binary inputs, and determines whether one number is greater than, less than or equal to the other number. The comparators are widely used in Central Processing Units (CPUs), Micro Controller Units (MCUs) which is a crucial data path element of image and signal processing architectures. In the last few years, the design of high-speed and low-power binary comparators has received a great deal of attention.

Several comparator designs are proposed to date include: High speed comparator [1], adder based comparator [3], Priority Encoder (PE) based comparator [2] [4**]**, BCL (Bitwise Competition Logic) comparator [6] etc…The comparator by Wang *et al*. [1] performs high speed comparison using All N Transistor (ANT). Power dissipation and area of this design is relatively large and is also not suitable for single cycle operation. Huang and Wang [2] and Lam and Tsui [4] designs use priority encoding algorithm for bit comparison. The elimination of long dynamic chain in these designs reduces delay compared to design in [1]. However the power dissipation of the PE based designs [2], [4] is high due to large switching, as the number of execution steps is more. Comparator by Stine and Schulte [3] uses hierarchical prefix tree structure which reduces delay and improves the scalability up to 2 bit comparison. However cascading of larger bit widths increase area and delay.

A MUX based comparator using Most Significant Bit (MSB) checking is proposed by Lam and Tsui [5]. Though the power dissipation of the Lam and Tsui design is low, the hardware complexity and delay are high. In another novel design, Kim and Yoo [6] used bitwise computation logic after pre encoding to find the first ‘1’ away from MSB. Perri and Corsonello [7] and Frustaci et al [8] proposed tree based structure for binary comparison. Though the computation speed of tree based structures in [7] and [8] is high, an implementation in static logic is not possible and their V_{dd}/V_t ratio is less. Krishna raj et al [9] designed a comparator using redundant binary signed digit number. The design demonstrates better delay reduction and is suitable for large operand comparison. Sharma et al.,[10] proposed a comparator design utilizing both PTL and CMOS logic . The hybrid comparator derives the advantages of both the logic and exhibit less power dissipation. Saleh Abdel-Hafeez [11] proposed scalable architecture for binary comparison. Though the design in [11] demonstrate better critical delay reduction, it is prone to high leakage power dissipation.

The comparator designs mentioned in the literature show better performance in terms of delay reduction but dissipate high dynamic power. On the other hand static designs exhibit less dynamic power dissipation compared to dynamic designs [12]. As the stack height of transistors grows exponentially with the number of variables in static logic, the designs mentioned in the literature are not suitable for static logic implementation. Also, higher stack height is less attractive in deep sub micrometer process, where the V_{dd}/V_t ratio is lower. So in this brief we propose a folded tree based comparator suitable for static logic implementation with reduced transistor stack height.

The rest of this paper is organized as follows: Section II gives a brief description of the existing comparator designs. Section III discusses the methodology involved in the proposed comparator. An illustration for the proposed comparator is given in section IV. Mathematical analysis of switching in the proposed binary comparator is given in section V. Experimental evaluation of the proposed comparator is discussed in section VI. An implementation of the proposed comparator in factorial calculator is discussed in section VII. A brief conclusion of the work done is given in Section VIII.

II. EXISTING TREE BASED BINARY COMPARATOR DESIGNS

Tree-based comparators are proposed by Perri and Corsonello [7] and Chuang et al [12] are suitable for static logic implementation which reduces dynamic power dissipation. The design in [7] uses CLA principle to find the greater of the two inputs. For 2 bit binary inputs A[1:0] and B[1:0], the addition of A and 2's complement of B generates a carry equal to "1" if A is greater than or equal to B and generates carry equal to "0" if B is greater than A. For cases where carry out is equal to '1" an additional examination is required to find whether A is greater than B or A is equal to B. This indeed requires an additional output which is an EXOR operation on inputs bit by bit. The comparator design in [12] also uses CLA technique where 2 indices "B_{BIG}" and "EQ" are generated to compare the inputs. "B_{BIG} , EQ" is "1,0" if B is greater than A, "B_{BIG} , EQ" is "0,0" if A is greater than B and "B_{BIG} , EQ" is "0,1" if B is equal to A. However the above tree based comparators occupy large area since for 64 bit binary inputs A [63:0] and B [63:0], they require 32 CLA units for carry generation and 32 equality checking blocks. This indeed occupies huge area and increases hardware complexity. To overcome this, we separate the binary inputs into groups called digit sets and perform comparison within digit sets starting from MSB. The checking of all the digit sets is done by a single pre-processing and encoding block where the different digit sets are time multiplexed on these units.

III. PROPOSED BINARY COMPARATOR

The proposed Area Efficient Folded Binary Comparator (AEFBC) consists of Pre-computation unit and Encoder block. The basic principle of AEFBC is to group the binary inputs into digit sets (digit size = word size/*m*, *m* being the number of digits formed). The digit sets are send to the precomputation unit starting from Most Significant Digit (MSD) to check for equality and the computations in precomputation unit are stopped at the first digit set which produces a "1" output. The corresponding digit set is send to the CLA encoder block to find the greatest of the two inputs. Thus, the proposed design avoids unnecessary checking of all the bits in the input. This reduces switching and huge dynamic power dissipation. Figure 1 shows the block diagram of the proposed AEFBC.

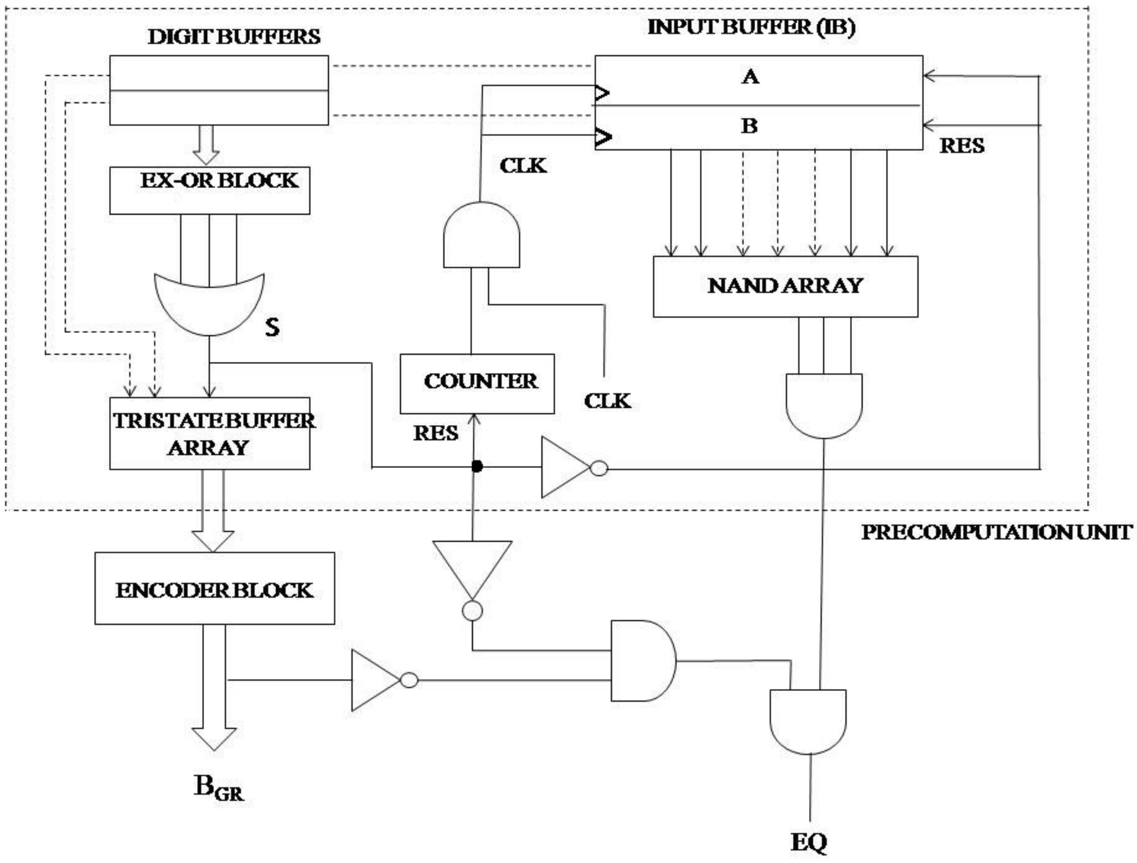

Fig.1 Block diagram of AEFBC

A. Pre – Computation Unit

The IB stores the two binary inputs. Based on the digit size the counter value is initialized. For each tick of the counter the bits are shifted into the Digit Buffer (DB) from the IB. The required number of bits are shifted into the DB when the counter output reaches "0".

The pre-computation unit performs EX-OR operation on bits in DB starting from Most Significant Bit (MSB). The EX-OR outputs are OR ed to find the equality within digits. If two digit sets are equal the ORed output will be zero and the next digit set will be passed to the DB for comparison. In case of unequal digit sets, ORed output will be "1", then the computations in the pre-computation block are stopped and the corresponding digit sets are send to the encoder block to determine the greatest of the two. On the other hand, if the output EX-OR is zero for all digit sets, then the input word is considered to be equal which can be realized by a "1" output at "EQ".

B. Encoder Block

The encoder block of the proposed comparator uses carry generation equation of CLA addition [12] to find the greatest among the two inputs. For a 2-bit digit (A_1A_0 and B_1B_0), comparison can be realized with the following Equation proposed by Chaung et al [12].

$$B_{GR} = \text{not}\left(A_1\right)B_1 + \text{not}\left(A_1 \text{ XOR } B_1\right)\left(\text{not}A_0 B_0\right) \qquad (1)$$

This is similar to the carry generation of a CLA adder given by

$$C_{out} = G + PC_{in} \qquad (2)$$

which can be written as

$$C_{out} = G_i + P_i G_{i-1} \qquad (3)$$

where C_{out} is equal to B_{GR} , Generate 'G_i' is equal to 'not(A_i)B_i' , 'P_i' is equal to 'not(A_i XOR B_i)'.

B_{GR} will be equal to "1" when the digit set $B_{DS}(B_iB_{i-1})$ is greater than digit set $A_{DS}(A_iA_{i-1})$ and B_{GR} will be equal to "0" if the digit set $B_{DS}(B_iB_{i-1})$ is less than digit set $A_{DS}(A_iA_{i-1})$.

For digits having more than two bits, the encoding operation is performed on 2 bits at a time starting from the MSB. The C_{out} s generated are ORed to find TC_{out}. If TC_{out} is equal to "1" then "B" is greater than "A" else "A" is greater than "B". The comparison done here is mainly based on radix-2 tree stucture since two bits are compared at a time.

IV. ILLUSTRATION OF THE PROPOSED METHOD

Consider two binary inputs A and B where

A = 10110110

B = 10111010

Case 1: 2 bit digit set grouping

Here A and B are grouped and compared as follows.

Digit set	A = 10 | 11 | 01 | 10
	B = 10 | 11 | 10 | 10	

Precomputation unit output "S"	0 0 1	
Encoder input	01 10	
Encoder output "B_{GR}"	1	B greater than A

Case 2: 4 bit digit set grouping

Here A and B are grouped and compared as follows.

Digit set	A = 1011 | 0110
	B = 1011 | 1010	

Precomputation unit output "S"	0 1	
Encoder input	01 | 10 10 | 10	
Encoder output "C_{out}"	1 0	
Encoder output "B_{GR}"	1	B greater than A

V. MATHEMATICAL ANALYSIS OF SWITCHING IN THE PROPOSED BINARY COMPARATOR

Let A [7:0] and B [7:0] be the inputs of the binary comparator, *n* – the number of bits in input (here n = 8), *d* – the number of bits in digit (here *d* = 2), i – the position of digit from MSB. The numbers of switching in the proposed and conventional comparator are analysed as follows.

Case i: 8 bit input with 1st digit not equal

If *MSD_1 of A: 00*

a) A: 00 A[5]A[4] A[3]A[2] A[1]A[0]
   B: 01 B[5]B[4] B[3]B[2] B[1]B[0]
   -Require 64 X 64 comparisons

b) A: 00 A[5]A[4] A[3]A[2] A[1]A[0]
   B: 10 B[5]B[4] B[3]B[2] B[1]B[0]
   -Require 64 X 64 comparisons

c) A: 00 A[5]A[4] A[3]A[2] A[1]A[0]
   B: 11 B[5]B[4] B[3]B[2] B[1]B[0]
   -Require 64 X 64 comparisons

If *MSD_1 of A : 01*

d) A: 01 A[5]A[4] A[3]A[2] A[1]A[0]

B: 00 B[5]B[4] B[3]B[2] B[1]B[0]

-Require 64 X 64 comparisons

e) A: 01 A[5]A[4] A[3]A[2] A[1]A[0]

B: 10 B[5]B[4] B[3]B[2] B[1]B[0]

-Require 64 X 64 comparisons

f) A: 01 A[5]A[4] A[3]A[2] A[1]A[0]

B: 11 B[5]B[4] B[3]B[2] B[1]B[0]

Require 64 X 64 comparisons

If *MSD_1 of A :10*

g) A: 10 A[5]A[4] A[3]A[2] A[1]A[0]

B: 00 B[5]B[4] B[3]B[2] B[1]B[0]

-Require 64 X 64 comparisons

h) A: 10 A[5]A[4] A[3]A[2] A[1]A[0]

B: 01 B[5]B[4] B[3]B[2] B[1]B[0]

-Require 64 X 64 comparisons

i) A: 10 A[5]A[4] A[3]A[2] A[1]A[0]

B: 11 B[5]B[4] B[3]B[2] B[1]B[0]

Require 64 X 64 comparisons

If *MSD_1 of A :11*

j) A: 11 A[5]A[4] A[3]A[2] A[1]A[0]

B: 00 B[5]B[4] B[3]B[2] B[1]B[0]

-Require 64 X 64 comparisons

k) A: 11 A[5]A[4] A[3]A[2] A[1]A[0]

B: 01 B[5]B[4] B[3]B[2] B[1]B[0]

-Require 64 X 64 comparisons

l) A: 11 A[5]A[4] A[3]A[2] A[1]A[0]

B: 10 B[5]B[4] B[3]B[2] B[1]B[0]

Require 64 X 64 comparisons

In total for d=2, d^2 -1 comparisons are required within digit set. Thus we have 2^d X (2^d-1) X 2^{n-Id} X 2^{n-Id} = 4 X 3 X 2^6 X 2^6 comparisons in conventional comparator which is reduced to 4 X 3 = 12 comparisons.

Case ii: 8 bit input with 1st digit equal but 2nd digit unequal

If *MSD_1 of A : 00 and MSD_1 of B : 00 / MSD_1 of A :11 and MSD_1 of B : 11 AND MSD_2 of A:00*

a) A: A[7]A[6] 00 A[3]A[2] A[1]A[0]

B: B[7]B[6] 01 B[3]B[2] B[1]B[0]

-Require 16 X 16 comparisons

b) A: A[7]A[6] 00 A[3]A[2] A[1]A[0]

B: B[7]B[6] 10 B[3]B[2] B[1]B[0]

-Require 16 X 16 comparisons

c) A: A[7]A[6] 00 A[3]A[2] A[1]A[0]

B: B[7]B[6] 11 B[3]B[2] B[1]B[0]

-Require 16 X 16 comparisons

-In total 3 X 16 X 16 comparisons.

Similarly

If *MSD_1 of A : 00 and MSD_1 of B : 00/ MSD_1 of A :11 and MSD_1 of B : 11ANDMSD$_2$ of A:01*

-Require 3 X 16 X 16 comparisons.

If *MSD_1 of A : 00 and MSD_1 of B : 00/ MSD_1 of A :11 and MSD_1 of B : 11 AND MSD_2 of A:10*

-Require 3 X 16 X 16 comparisons

If *MSD_1 of A : 00 and MSD_1 of B : 00 / MSD_1 of A :11 and MSD_1 of B : 11 AND MSD_2 of A:11*

-Require 3 X 16 X 16 comparisons

Total comparisons = 4 X 3 X 2^4 X 2^4 which is reduced to 12 comparisons.

Thus it can be seen that in case of unequal inputs the number of comparisons reduces at the rate of 2^{2id}.

VI. EXPERIMENTAL EVALUATION

The proposed AEFBC based on radix-2 tree-based structure has been designed using VHDL code and simulated using ALTERA QUARTUS II. Chuang et al. [12], Kim et al. [6] and Frustaci et al.[8] designs are used for comparison. The power, delay and area estimates of the proposed AEFBC and designs used for comparison for 16 bit input are shown in Table 1. It is seen that the AEFBC shows a delay reduction of 49.8%, 51.3% and 60.9% compared to Chuang et al. [12], Kim et al.[6] and Frustaci et al.[8] designs respectively. This is due to the elimination of long carry chain of checking bits from MSB to LSB in the proposed comparator. From the area estimates it can be seen that our comparator realizes the logic with fewer gate counts compared to all other previous designs. This is due to tree based realization of the proposed comparator. From the power dissipation estimates it can be seen that AEFBC exhibits minimum power dissipation compared to all other designs used for comparison.

TABLE I
Power Delay and Area Estimates of Proposed AEFBC and Existing Designs

Parameter		Proposed AEFBC		
		4 bit	**8 (4 bit grouping)**	**8 (2 bit grouping)**
Delay		7.272	10.381	8.913
Power		111.1	112.63	112.5
Area	AND	5	9	9
	XOR	5	12	12
	OR	6	5	6
	NOT	6	12	8

In addition, we have estimated the performance of our design for input operand: 4, 8 and shown in Table II. It is seen that delay and power dissipation reduces significantly for higher bits. Also it is seen that in case of 8 bit comparison, the power dissipation decreases with 2 bit grouping. This is due to, on an average the number of checking is less with 2 bit grouping compared to 4 bit grouping

TABLE III
Power Delay and Area Estimates of AEFBC for n = 4, 8

Parameter	**Proposed work**	**Chuang *et al.* [12]**	**Kim *et al.*[6]**	**Frustaci *et al.*[8]**
Delay (ns)	7.153	14.260	14.7	18.298
Power (mw)	114.12	115.18	116.5	116.20
Area(in gate count)	58	101	129	120

In case of gate count comparison, it is seen that the total number of gates is less for 2 bit grouping compared to 4 bit grouping. This is due to the increase in logic comparisons when the number of bits in grouping increase which necessitates more logic elements. The simulation waveforms of the proposed AEFBC for inputs A = 0001 and B = 0100 is shown in Figure 2. The input is split into two groups and as the MSD of A: “00” is smaller than

MSD of B: “01”, C_{out} is set to “1” which implies B_{GR} is “1”. The waveforms of 8 bit and 16 bit comparator are shown in Figure 3 and Figure 4 respectively.

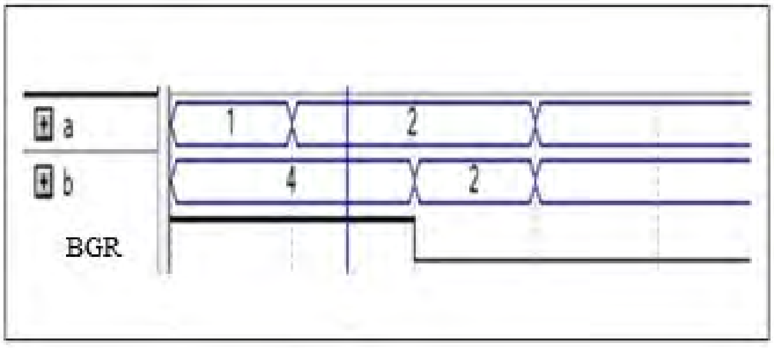

Fig.2. Simulation output of AEFBC for $n = 4$

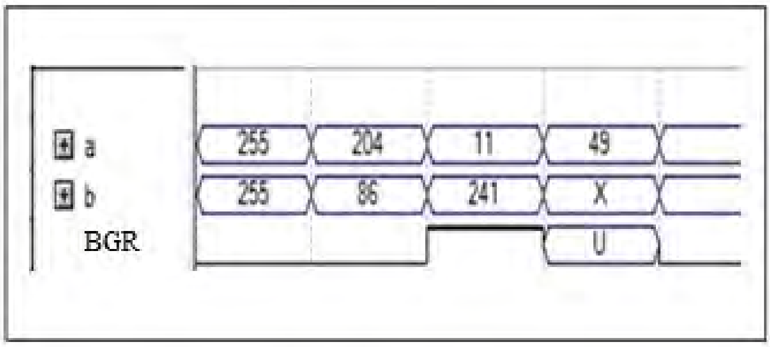

Fig.3. Simulation output of AEFBC for $n = 8$

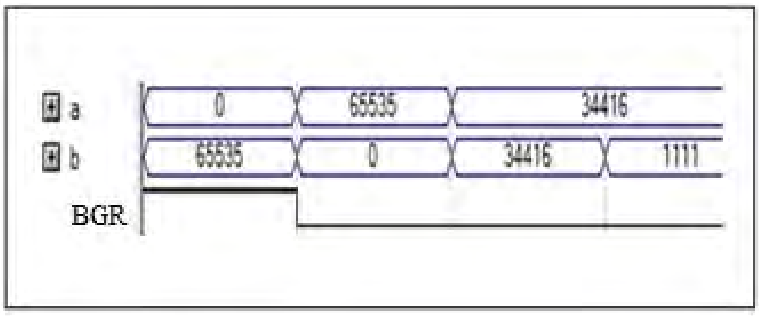

Fig. 4. Simulation output of AEFBC for $n = 16$

VII. IMPLEMENTATION OF PROPOSED COMPARATOR IN FACTORIAL CALCULATION

To verify the functionality of our proposed AEFBC, we have implemented it in Factorial calculator proposed by Saha et al [13]. We use Chuang et al design [12] for comparison. Area, delay and power dissipation are the parameters used for comparison. It is seen that the factorial calculator using our proposed comparator demonstrate better performance with an area reduction of 62%, delay of 16.66 % and power reduction of 3.01% compared to Chuang et al design implemented factorial calculator

VIII. CONCLUSION

ASIC implementation of a tree based binary comparator utilizing decision block and CLA technique is proposed in this brief. The inputs are split into groups and checking for equality starts from XOR operation of MSB group first. If MSB group is equal the checking proceeds to the next successive group towards LSB. On the other hand if any group is not equal the group is processed by an encoder block to find the greatest of the two using carry out signal of CLA addition. Experimental results demonstrate better performance of the proposed binary comparator when compared to the state of the art designs in terms of power and area reductions.

REFERENCES

[1] C.C. Wang, C.F. Wu, and K.C. Tsai, 1 GHz 64-bit high-speed comparator using ANT dynamic logic with two-phase clocking, IEEE *Proc.-Compute. Digit. Tech.*, Vol. 145, no. 6, pp. 433–436, 1998.

[2] C.H. Huang and J.S. Wang, High-performance and power-efficient CMOS comparators, *IEEE J. Solid-State Circuits*, Vol. 38, no. 2, pp. 254–262, 2003.

[3] J. E. Stine and M. J. Schulte, A combined two’s complement and floating-point comparator, *Proc. Int. Symp. Circuits Syst.*, Vol. 1, pp. 89–92, 2005.

[4] H.M. Lam and C.Y. Tsui, High-performance single clock cycle CMOS comparator, *Electron. Lett.*, Vol. 42, no. 2, pp. 75–77, 2006.

[5] H.M. Lam and C.Y. Tsui, A MUX-based high-performance “Single cycle CMOS comparator, *IEEE Trans. Circuits Syst. II, Exp. Briefs*,vol. 54, no. 7, pp. 591–595, 2007.

[6] J.Y. Kim and H.J. Yoo, Bitwise competition logic for compact “Digital comparator, in *Proc. IEEE Asian ASSCC*, 2007, pp. 59–62

[7] S. Perri and P. Corsonello, Fast low-cost implementation of single-Clock cycle binary comparator, *IEEE Trans. Circuits Syst. II, Exp. Briefs*, Vol. 55, no. 12, pp. 1239–1243, 2008

[8] F. Frustaci, S. Perri, M. Lanuzza, and P. Corsonello, A new low-Power high-speed single-clock-cycle binary comparator, Proc. *IEEE Int.Symp. Circuits Syst.*, pp. 317–320, 2010.

[9] Krishna Raj, Brijesh Kumar, Poornima Mittal, FPGA implementation and mask level CMOS layout design of redundant binary signed digit comparator IJCSNS, Vol.9 no.9, 2009.

[10] Geetanjali Sharma, Uma Nirmal, Yogesh Misra Low power 8-bit magnitude comparator with small transistor count using hybrid PTL/CMOS logic IJCEM, Vol. 12, 2011.

[11] Saleh Abdel-Hafeez, Scalable digital CMOS comparator using a parallel prefix tree, *Proc. IEEE Int.Symp. Circuits Syst.*, pp.107-115, 2012.

[12] Pierce Chuang, David Li, and Manoj Sachdev, A low-power high-performance single-cycle tree based 64-bit binary comparator IEEE Transactions On Circuits and Systems—II: Express Briefs, Vol.59, no. 2, 2012.

[13] P.Saha, A.Banerjee, A.Dandapat, P.Bhattacharyya ASIC design of a high speed low power circuit for factorial calculation using ancient Vedic mathematics Microelectronics Journal , Vol 42. pp.1343–1352. 2001.