# Power Efficient and High Speed Carry Skip Adder using Binary to Excess One Converter

Sanyukta Vijaykumar Chahande
Research Scholar (M.tech), Dept of ECE
Anjuman College of Engineering and Technology
Nagpur, India
*sanyuktavchahande@gmail.com*

Prof. Mohammad Nasiruddin
Associate. Professor & Head of Dept. of ECE
Anjuman College Of Engineering and Technology
Nagpur, India
*mn151819@gmail.com*

***Abstract***— The design of high-speed and low-power VLSI architectures need efficient arithmetic processing units, which are optimized for the performance parameters, namely, speed and power consumption. Adders are the key components in general purpose microprocessors and digital signal processors. As a result, it is very pertinent that its performance augers well for their speed performance. Additionally, the area is an essential factor which is to be taken into account in the design of fast adders. Towards this end, high-speed, low power and area efficient addition and multiplication have always been a fundamental requirement of high-performance processors and systems. The major speed limitation of adders arises from the huge carry propagation delay encountered in the conventional adder circuits, such as ripple carry adder and carry save adder. Observing that a carry may skip any addition stages on certain addend and augend bit values, researchers developed the carry-skip technique to speed up addition in the carry-ripple adder. Using a multilevel structure, carry-skip logic determines whether a carry entering one block may skip the next group of blocks. Because multilevel skip logic introduces longer delays, Therefore, in this paper we examine The basic idea of this work is to use Binary to Excess- 1 converter (BEC) instead of RCA with Cin=1 in conventional CSkA in order to reduce the area and power. BEC uses less number of logic gates than N-bit full adder.

***Key Words:*** *Carry Skip Adder, Carry Skip Logic, Binary to Excess-1 converter*

*****

## I. Introduction

High speed and low power adder circuits have become very important in VLSI industry. Adder circuit is one of the most important building block in DSP processor. Adder is the a vital component in most of the arithmetic unit. Addition is a fundamental operation for any digital system, DSP or control system. The fast operation of a digital system is having great influenced by the performance of the resident adders. Adders plays important Component in digital systems because of the more number for use in other basic digital operations such as subtraction, multiplication and division. Hence, the improving performance of the digital adder increases the execution of various binary operations in a circuit consisting of different blocks. The appearance of the digital circuit block is gauged by analyzing its power dissipation, layout area and its operating speed. There are many works on the subject of optimizing the speed and power of these units, which has been reported. Obviously, it is extremely possible to achieve higher speeds at low-power and energy consumptions, which is one of the challenges for the designers of general purpose processors. The carry skip adder is one of the fastest adders used in many digital processors to perform arithmetic operations. A carry-skip adder consists of a simple ripple carry-adder with a special speed up carry chain called a **skip chain**. This chain defines the distribution of ripple carry blocks, which compose the skip adder. The carry skip adder comes under the category of a by-pass adder and it uses a ripple carry adder for an adder implementation. The formation of carry skip adder block is attained by improving a worst-case delay. The carry skip adder has a critical path, that passes through all adders and stops at the sum bit but actually it starts at first full adder. This adder is an efficient one according to its area usage and power consumption. The structure of 4 bit carry skip adder is shown in fig.1.

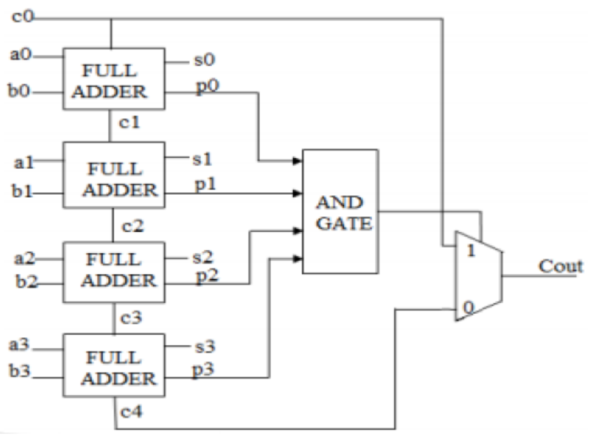


**Fig.1. Four bit carry skip adder.**

### 1.1 Basic mechanism of CSkA

The addition of two binary digits at stage i, where i not equal to 0, of the ripple carry adder depends on the carry in, ***Ci*** , which in reality is the carry out, ***Ci-1***, of the previous stage. Therefore, in order to calculate the sum and the carry out,

***Ci+1*** , of stage i, it is imperative that the carry in, ***Ci***, be known in advance.

**Pi = Ai** XOr **Bi** Equ. 1 --carry propagate of ith stage

**Si = Pi** XOr **Ci** Equ. 2 --sum of ith stage

**Ci+1 = AiBi + PiCi** Equ. 3 --carry out of ith stage

Supposing that ***Ai*** = ***Bi***, then ***Pi*** in equation 1 would become zero (equation 4). This would make ***Ci+1*** to depend only on the inputs ***Ai*** and ***Bi***, without needing to know the value of ***Ci***.

**Ai = Bi** then **Pi = 0** Equ. 4 --from #Equation 1

If **Ai = Bi = 0** then **Ci+1 = AiBi = 0** --from equation 3

If **Ai = Bi = 1** then **Ci+1 = AiBi = 1** --from equation 3

Therefore, if Equation 4 is true then the carry out, ***Ci+1***, will be one if ***Ai*** = ***Bi*** = ***1*** or zero if ***Ai*** = ***Bi*** = ***0***. Hence we can compute the carry out at any stage of the addition provided equation 4 holds. These findings would enable us to build an adder whose average time of computation would be proportional to the longest chains of zeros and of different digits of A and B.

Alternatively, given two binary strings of numbers, such as the example below, it is very likely that we may encounter large chains of consecutive bits (block 2) where ***Ai*** not equal to ***Bi***. In order to deal with this scenario we must reanalyze equation 3 carefully.

**Ai** not equal to **Bi** then **Pi = 1** Equ. 5 --from Equation 1

If **Ai** not equal to **Bi** then **Ci+1 = Ci** --from Equation 3

In the case of comparing two bits of opposite value, the carry out at that particular stage, will simply be equivalent to the carry in. Hence we can simply propagate the carry to the next stage without having to wait for the sum to be calculated.

**Two Random Bit Strings:**

| | | | | |
|---|---|---|---|---|
| **A** | 10100 | 01011 | 10100 | 01011 |
| **B** | 01101 | 10100 | 01010 | 01100 |
| | block 3 | block 2 | block 1 | block 0 |

## 1.2 Chain Mechanism

In order to take advantage of the last property, we can design an adder that is divided into blocks, as shown in Fig. 2, where a special purpose circuit can compare the two binary strings inside each block and determine if they are equal or not. In the latter case the carry entering the block will simply be propagated to the next block and if this is the case all the carry inputs to the bit positions in that block are all either 0's or 1's depending on the carry in into the block. Should only one pair of bits (***Ai*** and ***Bi***) inside a block be equal then the carry skip mechanism would be unable to skip the block. In the extreme case, although still likely, that there exist one such case, where ***Ai*** = ***Bi***, in each block, then no block is skipped but a carry would be generated in each block instead. Two strings of binary numbers to be added are divided into blocks of equal length. In each cell within a block both bits are compared for un-equivalence. This is done by Exclusive ORing each individual cell (parallel operation and already present in the full adder) producing a comparison string. Next the comparison string is ANDed within itself in a domino fashion. This process ensures that the comparison of each and all cells was indeed unequal and we can therefore proceed to propagate the carry to the next block. A MUX is responsible for selecting a **generated carry** or a **propagated** (previous) **carry** with its selection line being the output of the comparison circuit just described. If for each cell in the block ***Ai ≠ Bi*** then we say that a carry can skip over the block otherwise if ***Ai*** = ***Bi*** we shall say that the carry must be generated in the block.

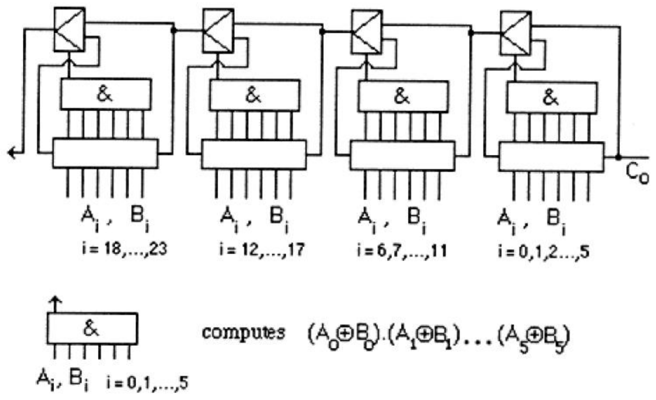

&
$C_0$
$A_i$, $B_i$
i = 18,...,23
i = 12,...,17
i = 6,7,...,11
i = 0,1,2...,5
computes $(A_0 \oplus B_0).(A_1 \oplus B_1)...(A_5 \oplus B_5)$
$A_i$, $B_i$ i = 0,1,...,5

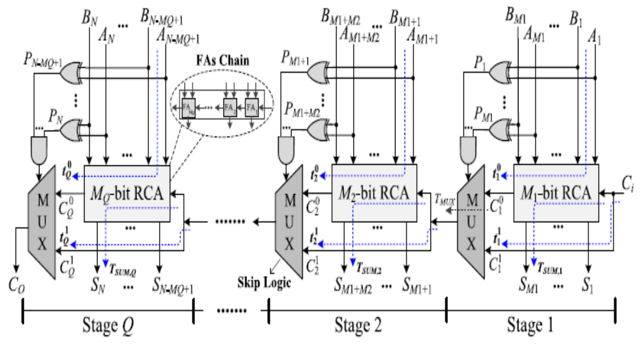
**Fig 2: Carry Chain Skip Mechanism**

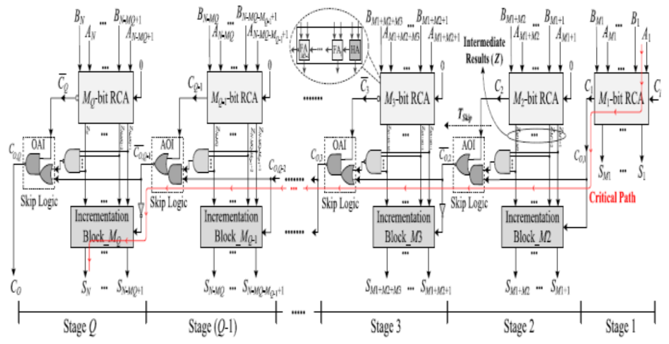
## II. Prior Work

As this paper is on the CSKA structure, first the related work to this adder are reviewed and then carry skip adder using Binary to Excess-1 converter structures are discussed.

The conventional structure of the CSKA consists of stages containing chain of full adders (FAs) (RCA block) and 2:1 multiplexer (carry skip logic). The RCA blocks are
connected to each other through 2:1 multiplexers, which can be placed into one or more level structures. Many methods have been suggested for finding the optimum number of the FAs [18]–[26]. The techniques presented in [19]–[24] make use of VSSs to minimize the delay of adders based on a single level carry skip logic. In [25], some methods to increase the speed of the multilevel CSKAs are proposed. The techniques, however, cause area and power increase considerably and less regular layout. In addition, to lower the propagation delay of the adder, in each stage, the carry look-ahead logics were utilized. Again, it had a complex layout as well as large power consumption and area usage. In addition, the design approach, which was presented only for the 32-bit adder, was not general to be applied for structures with different bits lengths. Based on the discussion presented above, it is concluded that by reducing the delay of the skip logic, one may lower the propagation delay of the CSKA significantly[28].Hence, a new CSkA structure was presented in which the multiplexers were replaced by AOI/OAI logic in this structure as logic gates

were used instead of MUX which results into less power consumption and there by increasing the speed of the adder.





## III. PROPOSED STRUCTURE

This structure is based on a concept of using Binary to Excess on Converter. The basic idea of this of this work is to use BEC instead of AOI/OAI logic in order to reduce the power consumption and thereby increase the speed of adder as it requires less number of gates. Excess-3 binary coded decimal or Stibitz code, also called biased representation of Excess-N, is a complementary BCD cod and numeral system. It is a way to represent values with a balanced number of positive and negative numbers using a pre-specified number N as a biasing value. It is a non- weighted code. XS-3, numbers are represented as decimal digits, and each digit is represented by four bits as the digit value plus 3 (the "excess" amount):
1. The smallest binary number represents the smallest value. (i.e. 0 − Excess Value).

2. The greatest binary number represents the largest value. (i.e. 2 N+1 − Excess Value − 1).

The primary advantage of XS-3 coding over non-biased coding is that a decimal number can be nines' complemented (for subtraction) as easily as a binary number can be ones' complemented (to invert all bits)[19] As stated earlier the main idea of this work is to use BEC instead of the RCA with in order to reduce the area and power consumption of the regular CSLA. To replace the n-bit RCA, an n+1 bit BEC is required. A structure and the function table of a 4-b BEC are shown in Fig. 3 respectively.

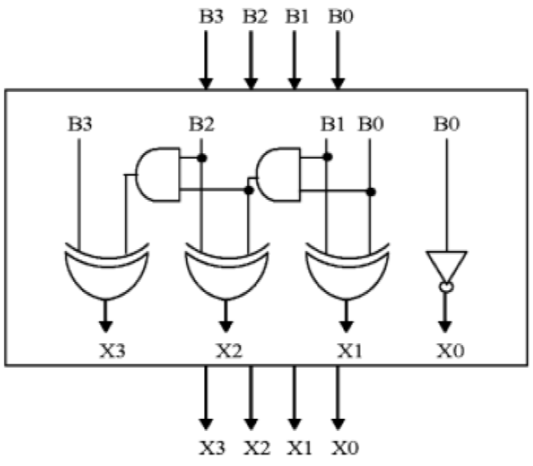


**Fig 3: 4 bit BEC**

**Table -1:** Truth Table of 4 bit binary to Excess One Converter

| Binary Logic $B_0\,B_1\,B_2\,B_3$ | Excess-1 Logic $X_0\,X_1\,X_2\,X_3$ |
|---|---|
| 0000 | 0001 |
| 0001 | 0010 |
| 0010 | 0011 |
| 0011 | 0100 |
| 0100 | 0101 |
| 0101 | 0110 |
| 0110 | 0111 |
| 0111 | 1000 |
| 1000 | 1001 |
| 1001 | 1010 |
| 1010 | 1011 |
| 1011 | 1100 |
| 1100 | 1101 |
| 1101 | 1110 |
| 1110 | 1111 |
| 1111 | 0000 |

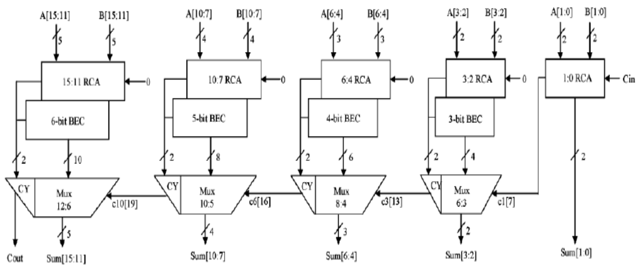
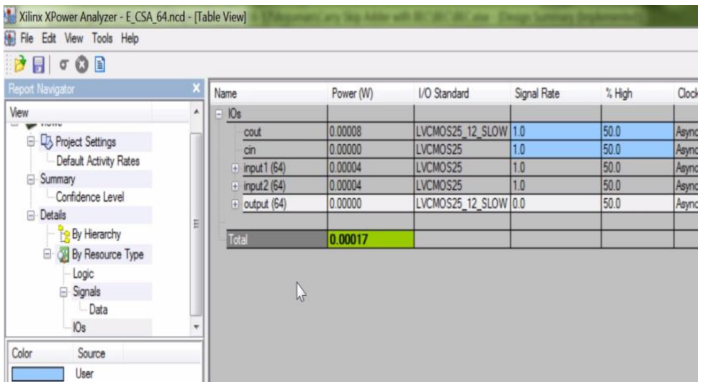


**Fig 4:CSkA using BEC**


Xilinx XPower Analyzer - E_CSA_64.ncd - [Table View]

| Name | Power (W) | I/O Standard | Signal Rate | % High | Clock |
|---|---|---|---|---|---|
| IOs | | | | | |
| cout | 0.00008 | LVCMOS25_12_SLOW | 1.0 | 50.0 | Async |
| cin | 0.00000 | LVCMOS25 | 1.0 | 50.0 | Async |
| input1 (64) | 0.00004 | LVCMOS25 | 1.0 | 50.0 | Async |
| input2 (64) | 0.00004 | LVCMOS25 | 1.0 | 50.0 | Async |
| output (64) | 0.00000 | LVCMOS25_12_SLOW | 0.0 | 50.0 | Async |
| Total | 0.00017 | | | | |

## IV. CONCLUSIONS

A simple approach is proposed in this project to increase the speed and reduce the power consumption of CSkA architecture. The reduced number of gates of this project offers the great advantage in the reduction of power consumption and also the increase the speed. The compared results show that the modified CSkA has a slightly larger area, but the power consumption of the modified CSkA are significantly reduced. The modified CSLA architecture is therefore, low power, high speed simple and efficient for VLSI hardware implementation.. It can be observed that for the modified CSkA is 79% power efficient when compared to the regular CSkA.

| **Design** | **Area** | **Power** | **Delay** |
|---|---|---|---|
| **Normal 64 bit CSkA** | **112** | **0.00084** | **1.48 ns** |
| **CSkA with BEC 64 bit** | **212** | **0.00017** | **1.48ns** |

## REFERENCES

[1] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. Natick, MA, USA: A K Peters, Ltd., 2002.

[2] R. Zlatanovici, S. Kao, and B. Nikolic, "Energy–delay optimization of 64-bit carry-lookahead adders with a 240 ps 90 nm CMOS design example," *IEEE J. Solid-State Circuits*, vol. 44, no. 2, pp. 569–583, Feb. 2009.

[3] S. K. Mathew, M. A. Anders, B. Bloechel, T. Nguyen, R. K. Krishnamurthy, and S. Borkar, "A 4-GHz 300-mW 64-bit integer execution ALU with dual supply voltages in 90-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 44–51, Jan. 2005.

[4] V. G. Oklobdzija, B. R. Zeydel, H. Q. Dao, S. Mathew, and R. Krishnamurthy, "Comparison of high-performance VLSI adders in the energy-delay space," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 6, pp. 754–758, Jun. 2005.

[5] B. Ramkumar and H. M. Kittur, "Low-power and area-efficient carry select adder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 2, pp. 371–375, Feb. 2012.

[6] M. Vratonjic, B. R. Zeydel, and V. G. Oklobdzija, "Low- and ultra low-power arithmetic units: Design and comparison," in *Proc. IEEE Int. Conf. Comput. Design, VLSI Comput. Process. (ICCD)*, Oct. 2005, pp. 249–252.

[7] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 689–702, Oct. 1996.

[8] Y. He and C.-H. Chang, "A power-delay efficient hybrid carrylookahead/ carry-select based redundant binary to two's complement converter," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 1, pp. 336–346, Feb. 2008.

[9] C.-H. Chang, J. Gu, and M. Zhang, "A review of 0.18 $\mu$m full adder performances for tree structured arithmetic circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 6, pp. 686–695, Jun. 2005.

[10] D. Markovic, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow-power design in near-threshold region," *Proc. IEEE*, vol. 98, no. 2, pp. 237–252, Feb. 2010.

[11] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," *Proc. IEEE*, vol. 98, no. 2,pp. 253–266, Feb. 2010.

[12] S. Jain *et al.*, "A 280 mV-to-1.2 V wide-operating-range IA-32 processor in 32 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 2012, pp. 66–68.

[13] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. dissertation, Dept. Inf. Technol. Elect. Eng., Swiss Federal Inst. Technol. (ETH), Zürich, Switzerland, 1998.

*[14]* D. Harris, "A taxonomy of parallel prefix networks," in *Proc. IEEE Conf. Rec. 37th Asilomar Conf. Signals, Syst., Comput.*, vol. 2. Nov. 2003, pp. 2213–2217.

[15] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comput.*, vol. C-22, no. 8, pp. 786–793, Aug. 1973.

[16] V. G. Oklobdzija, B. R. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-delay estimation technique for high performance microprocessor VLSI adders," in *Proc. 16th IEEE Symp. Comput. Arithmetic*, Jun. 2003, pp. 272–279.

[17] M. Lehman and N. Burla, "Skip techniques for high-speed carrypropagation in binary arithmetic units," *IRE Trans. Electron. Comput.*, vol. EC-10, no. 4, pp. 691–698, Dec. 1961.

[18] K. Chirca *et al.*, “A static low-power, high-performance 32-bit carry skip adder,” in *Proc. Euromicro Symp. Digit. Syst. Design (DSD)*, Aug./Sep. 2004, pp. 615–619.

[19] 19.K Prakash Rao, A Uday Kumar” Design of a low power 16. Bit CSLA using Binary to Excess one Converter”, vol no.5 special issue(1), feb 2016

[20] S. Majerski, “On determination of optimal distributions of carry skips in adders,” *IEEE Trans. Electron. Comput.*, vol. EC-16, no. 1, pp. 45–58,

[21] Feb. 1967.

[22] A. Guyot, B. Hochet, and J.-M. Muller, “A way to build efficient carryskip adders,” *IEEE Trans. Comput.*, vol. C-36, no. 10, pp. 1144–1152, Oct. 1987.

*[23]* S. Turrini, “Optimal group distribution in carry-skip adders,” in *Proc. 9th IEEE Symp. Comput. Arithmetic*, Sep. 1989, pp. 96–103.

[24] P. K. Chan, M. D. F. Schlag, C. D. Thomborson, and V. G. Oklobdzija, “Delay optimization of carry-skip adders and block carry-lookahead adders using multidimensional dynamic programming,” *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 920–930, Aug. 1992.

*[25]* V. Kantabutra, “Designing optimum one-level carry-skip adders,” *IEEE Trans. Comput.*, vol. 42, no. 6, pp. 759–764, Jun. 1993.

[26] V. Kantabutra, “Accelerated two-level carry-skip adders—A type of very fast adders,” *IEEE Trans. Comput.*, vol. 42, no. 11, pp. 1389–1393, Nov. 1993.

[27] S. Jia *et al.*, “Static CMOS implementation of logarithmic skip adder,” in *Proc. IEEE Conf. Electron Devices Solid-State Circuits*, Dec. 2003, pp. 509–512.

[28] H. Suzuki, W. Jeong, and K. Roy, “Low power adder with adaptivesupply voltage,” in *Proc. 21st Int. Conf. Comput. Design*, Oct. 2003, pp. 103–106.

[29] 28.Milad Bahadori, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram, “High-Speed and Energy-Efficient Carry Skip Adder Operating Under a Wide Range of Supply Voltage Levels”, 1063-8210 © 2015 IEEE.