INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 4, Issue 4 AUG 2015

Design And Implementation Of 128-bit Carry Select Adder For High Speed Applications

A. MANASA[(1)] U. ASFIA BEGUM[(2)]

M.Tech(Scholar-DECS),K.V.Subba Reddy College Of Engineering For Women[(1)]

M.Tech, Asst.prof.,K.V.Subba Reddy College Of Engineering For Women[(2)]

***A b s t r a c t*—carry select adder increases the speed of the addition process by reducing the carry propagation time to the minimum commensurate with economical circuit design .In this brief, the logic operations involved in conventional carry select adder (CSLA) and binary to excess-1 converter(BEC)-based CSLA are analyzed to study the data dependence and to identify redundant logic operations. We have eliminated all the redundant logic operations present in the conventional CSLA and proposed a new logic formulation for CSLA. In the proposed scheme, the carry select (CS) operation is scheduled before the calculation of *fi n a l - s u m*, which is different from the conventional approach. Bit patterns of two anticipating carry words (corresponding to c_{in} = 0 and 1) and fixed c_{in} bits are used for logic optimization of CS and generation units. An efficient CSLA design is obtained using optimized logic units. The proposed CSLA design involves significantly less area and delay than the recently proposed BEC-based CSLA. Due to the small carry-output delay, the proposed CSLA design is a good candidate for square-root (SQRT) CSLA. A theoretical estimate shows that the proposed SQRT-CSLA involves nearly 35% less area–delay–product (ADP) than the BEC-based SQRT-CSLA, which is best among the existing SQRT-CSLA designs, on average, for different bit-widths. The application-specified integrated circuit (ASIC) synthesis result shows that the BEC-based SQRT-CSLA design involves 48% more ADP and consumes 50% more energy than the proposed SQRT-CSLA, on average, for different bit-widths.**

***Index Terms*—Adder, arithmetic unit, low-power design.**

I. INTRODUCTION

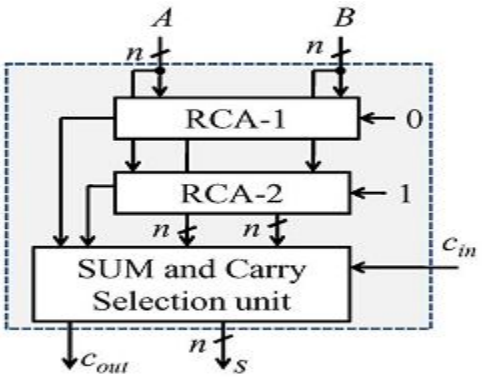
LOW-POWER, area-efficient, and high-performance VLSI systems are increasingly used in portable and mobile devices, multi standard wireless receivers, and biomedical instrumentation [1], [2]. An adder is the main component of an arithmetic unit. A complex digital signal processing (DSP) system involves several adders.

An efficient adder design essentially improves the performance of a complex DSP system. A ripple carry adder (RCA) uses a simple design, but carry propagation delay (CPD) is the main concern in this adder. Carry look-ahead and carry select (CS) methods have been suggested to reduce the CPD of adders. A conventional carry select adder (CSLA) is an RCA–RCA configuration that generates a pair of *sum* words and *output carry* bits corresponding the anticipated input-carry (c_{in} = 0 and 1) and selects one out of each pair for *final-sum* and *final output-carry* [3]. A conventional CSLA has less CPD than an RCA, but the design is not attractive since it uses a dual RCA. Few attempts have been made to avoid dual use of RCA in CSLA design. Kim and Kim [4] used one RCA and one add-one circuit instead of two RCAs, where the add-one circuit is implemented using a multiplexer (MUX). He *et al.* [5] proposed a square-root (SQRT)-CSLA to implement large bit-width adders with less delay. In a SQRT CSLA, CSLAs with increasing size are connected in a cascading structure. The main objective of SQRT-CSLA design is to provide a parallel path for carry propagation that helps to reduce the overall adder delay. Ramkumar and Kittur [6] suggested a binary to BEC-based CSLA. The BEC-based CSLA involves less logic resources than the conventional CSLA, but it has marginally higher delay. A CSLA based on common Boolean logic (CBL) is also proposed in [7] and [8]. The CBL-based CSLA of [7] involves significantly less logic resource than the conventional CSLA but it has longer CPD, which is almost equal to that of the RCA. To overcome this problem, a SQRT-CSLA based on CBL was proposed in [8]. However, the CBL-based SQRTCSLA design of [8]

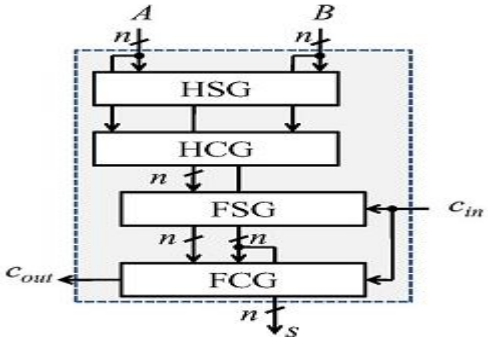
requires more logic resource and delay than the BEC-based SQRT-CSLA of [6]. We observe that logic optimization largely depends on availability of redundant operations in the formulation, whereas adder delay mainly depends on data dependence. In the existing designs, logic is optimized without giving any consideration to the data dependence. In this brief, we made an analysis on logic operations involved in conventional and BEC-based CSLAs to study the data dependence and to identify redundant logic operations. Based on this analysis, we have proposed a logic formulation for the CSLA. The main contribution in this brief are logic formulation based on data dependence and optimized carry generator (CG) and CS design.

II. LOGIC FORMULATION

The CSLA has two units: 1) the *sum* and *carry* generator unit (SCG) and 2) the *sum* and *carry* selection unit [9]. The SCG unit consumes most of the logic resources of CSLA and significantly contributes to the critical path. Different logic designs have been suggested for efficient implementation of the SCG unit. We made a study of the logic designs suggested for the SCG unit of conventional and BEC-based CSLAs of [6] by suitable logic expressions. The main objective of this study is to identify redundant logic operations and data dependence. Accordingly, we remove all redundant logic operations and sequence logic operations based on their data dependence.



(a)



(b)

Fig. 1. (a) Conventional CSLA; *n* is the input operand bit-width. (b) The logic operations of the RCA is shown in split form, where HSG, HCG, FSG, and FCG represent half-sum generation, half-carry generation, full-sum generation, and full-carry generation, respectively.

A.Logic Expressions of the SCG Unit of the Conventional CSLA

As shown in Fig. 1(a), the SCG unit of the conventional CSLA [3] is composed of two *n*-bit RCAs, where *n* is the adder bit-width. The logic operation of the *n*-bit RCA is performed in four stages: 1) *half-sum* generation (HSG); 2) *half-carry* generation (HCG); 3) *full-sum* generation (FSG); and 4) *fullcarry* generation (FCG). Suppose two *n*-bit operands are added in the conventional CSLA, then RCA-1 and RCA-2 generate *n*-bit *sum* (s_0 and s_1) and output-carry (c_{0out} and c_{1out}) corresponding to input-carry (c_{in} = 0 and c_{in} = 1), respectively. Logic expressions of RCA-1 and RCA-2 of the SCG unit of then-bit CSLA are given as

$$s^0_0(i) = A(i) \oplus B(i) \quad c_0^0(i) = A(i) \cdot B(i) \quad (1a)$$

$$s^0_1(i) = s^0_0(i) \oplus c^0_1(i-1) \quad (1b)$$

$$c_0^1(i) = c_0(i) + s_0(i) \cdot c^0_1(i-1) \; c_{0out} = c_0^1(n-1) \quad (1c)$$

$$s_{10}(i) = A(i) \oplus B(i) \quad c_{10}(i) = A(i) \cdot B(i) \quad (2a)$$

$$s_1^1(i) = s^1_0(i) \oplus c_1^1(i-1) \quad (2b)$$

$$c_1^1(i) = c^1_0(i) + s^1_0(i) \cdot c^1_1(i-1) \; c^1_{out} = c^1_1(n-1) \quad (2c)$$

where $c^0_1(-1) = 0$, $c^1_1(-1) = 1$, and $0 \le i \le n - 1$. As shown in (1a)–(1c) and (2a)–(2c), the logic expression of $\{s_{0}(i), c_{0}(i)\}$ is identical to that of $\{s_{10}(i), c_{10}(i)\}$. These redundant logic operations can be removed to have an optimized design for RCA-2, in which the HSG and HCG of RCA-1 is shared to construct RCA-2. Based on this, [4] and [5] have used an add-one circuit instead of RCA-2 in the CSLA, in which a BEC circuit is used in [6] for the same purpose. Since the BEC-based CSLA offers the best area–delay–power efficiency among the existing CSLAs, we discuss here the logic expressions of the SCG unit of the BEC-based CSLA as well.

B. Logic Expression of the SCG Unit of the BEC-BasedCSLA

As shown in Fig. 2, the RCA calculates n-bit *sum* s^0_1 and c^0_{out} corresponding to $c_{in} = 0$. The BEC unit receives s^0_1 and c^0_{out} from the RCA and generates $(n + 1)$-bit excess-1 code. The most significant bit (MSB) of BEC represents c^1_{out}, in which n least significant bits (LSBs) represent s^1_1. The logic expressions of the RCA are the same as those given in (1a)–(1c). The logic expressions of the BEC unit of the n-bit BEC-based CSLA are given as

$$s^1_1(0) = s^0_1(0)\ c^1_1(0) = s^0_1(0) \quad (3a)$$

$$s^1_1(i) = s^0_1(i) \oplus c^1_1(i-1) \quad (3b)$$

$$c^1_1(i) = s^0_1(i) \cdot c^1_1(i-1) \quad (3c)$$

$$c^1_{out} = c^0_1(n-1) \oplus c^1_1(n-1) \quad (3d)$$

for $1 \le i \le n - 1$.We can find from (1a)–(1c) and (3a)–(3d) that, in the case of the BEC-based CSLA, c_1 depends on s_{01}, which otherwise has no dependence on s_{01} in the case of the conventional CSLA. The BEC method therefore increases data dependence in the CSLA. We have considered logic expressions of the conventional CSLA and made a further study on the data dependence to find an optimized logic expression for the CSLA

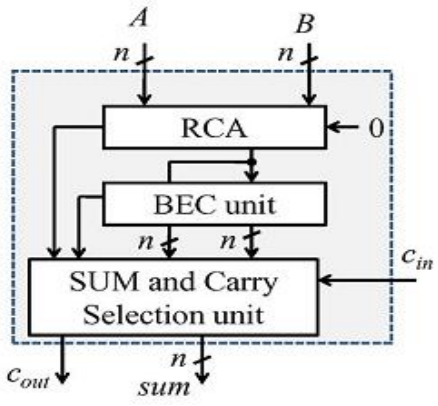


Fig. 2. Structure of the BEC-based CSLA; n is the input operand bit-width.

It is interesting to note from (1a)–(1c) and (2a)–(2c) that logic expressions of s_{01} and s_1 are identical except the termsc^0_1 and c^1_1 since $(s^0_0 = s^1_0 = s_0)$. In addition, we find that c^0_1 and c^1_1 depend on $\{s_0, c_0, c_{in}\}$, where $c_0 = c_0 = c_{10}$. Since c^0_1 and c^1_1 have no dependence on s^0_1 and s^1_1, the logic operation of c^0_1 and c^1_1 can be scheduled before s^0_1 and s^1_1, and the select unit can select one from the set (s_{01}, s_1) for the *final-sum* of the CSLA. We find that a significant amount of logic resource is spent for calculating $\{s_{01}, s_1\}$, and it is not an efficient approach to reject one sum-word after the calculation. Instead, one can select the required carry word from the anticipated carry words { c_0 and c_1} to calculate the *final-sum*. The selected carry word is added with the *half-sum* (s_0) to generate the *final-sum* (s). Using this method, one can have three design advantages:1) Calculation of s_{01} is avoided in the SCG unit; 2) the n-bit select unit is required instead of the $(n + 1)$ bit; and 3) small output-carry delay. All these features result in an area–delay and energy-efficient design for the CSLA. We have removed all the redundant logic operations of (1a)–(1c) and (2a)–(2c) and rearranged logic expressions of (1a)–(1c) and (2a)–(2c) based on their dependence. The proposed logic formulation for the CSLA is given as

$$s_0(i) = A(i) \oplus B(i)\ c_0(i) = A(i) \cdot B(i) \quad (4a)$$

$$c^0_1(i) = c^0_1(i-1) \cdot s_0(i) + c_0(i) \text{ for } c^0_1(0) = 0 \quad (4b)$$

$$c^1_1(i) = c^1_1(i-1) \cdot s_0(i) + c_0(i) \text{ for } c^1_1(0) = 1$$

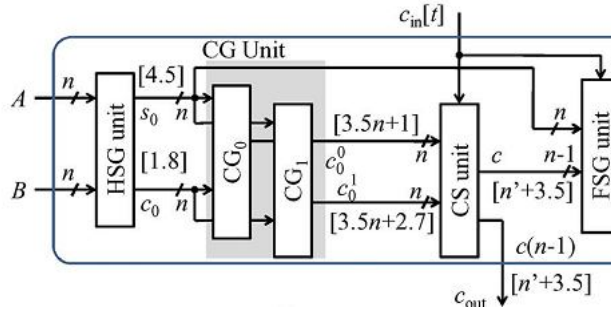
(4c)

$c(i) = c^0_1(i)$ if $(c_{in} = 0)$ (4d)

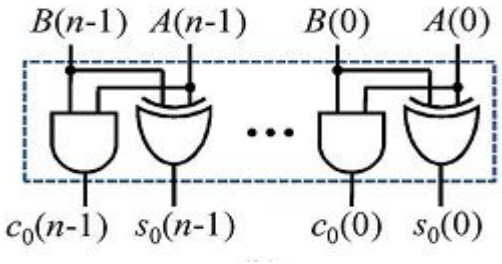
$c(i) = c^1_1(i)$ if $(c_{in} = 1)$ (4e)

$c_{out} = c(n - 1)$ (4f)

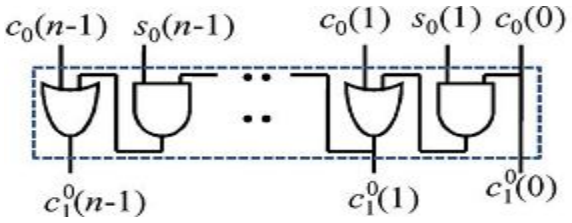
$s(0) = s_0(0) \oplus c_{in}\, s(i) = s_0(i) \oplus c(i - 1).$ (4g)



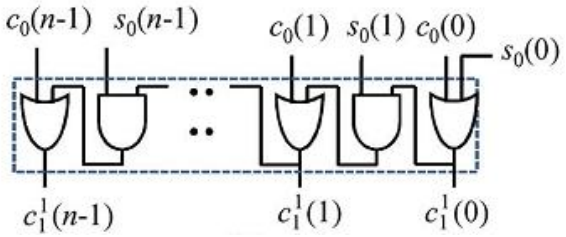
(a)



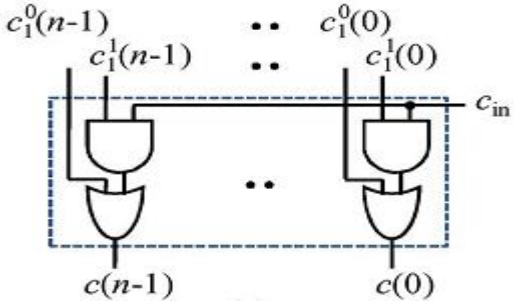
(b)



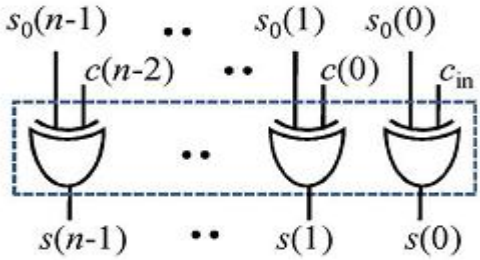
(c)



(d)



(e)



(e)

Fig. 3. (a) Proposed CS adder design, where n is the input operand bit-width, and [$*$] represents delay (in the unit of inverter delay), $n = \max(t, 3.5n + 2.7)$. (b) Gate-level design of the HSG. (c) Gate-level optimized design of (CG 0) for input-carry = 0. (d) Gate-level optimized design of (CG 1) for input-carry = 1. (e) Gate-level design of the CS unit. (f) Gate-level design of the final-sum generation (FSG) unit.

III. PROPOSED ADDER DESIGN

The proposed CSLA is based on the logic formulation given in (4a)–(4g), and its structure is shown in Fig. 3(a). It consists of one HSG unit, one FSG unit, one CG unit, and one CS unit. The CG unit is composed of two CGs (CG_0 and CG_1) corresponding to input-carry '0' and '1'. The HSG receives two n-bit operands (A and B) and generate *half-sum* word s_0 and *half-carry* word c_0 of width n bits each. Both CG_0 and CG_1 receive s_0 and c_0 from the HSG unit and generate two n-bit full-carry words c^0_1 and c^1_1 corresponding to input-carry '0' and '1', respectively.

The logic diagram of the HSG unit is shown in Fig. 3(b). The logic circuits of CG_0 and CG_1 are optimized to take advantage of the fixed input-carry bits. The

optimized designs of CG_0 and CG_1 are shown in Fig. 3(c) and (d), respectively.

The CS unit selects one final carry word from the two carry words available at its input line using the control signal c_{in}. It selects c_{01} when $c_{in} = 0$; otherwise, it selects c_1. The CS unit can be implemented using an n-bit 2-to-l MUX. However, we find from the truth table of the CS unit that carry words c_{01} and c_{11} follow a specific bit pattern. If $c_{01}(i)$ = '1', then $c_1(i) = 1$, irrespective of $s_0(i)$ and $c_0(i)$, for $0 \leq i \leq n - 1$. This feature is used for logic optimization of the CS unit. The optimized design of the CS unit is shown in Fig. 3(e), which is composed of n AND–OR gates. The final carry word c is obtained from the CS unit. The MSB of c is sent to output as c_{out}, and $(n - 1)$ LSBs are XORed with $(n - 1)$ MSBs of *half-sum* (s_0) in the FSG [shown in Fig. 3(f)] to obtain $(n - 1)$ MSBs of *final-sum* (s). The LSB of s_0 is XORed with c_{in} to obtain the LSB of s.

III. Performance Comparison

A. Area–Delay Estimation Method

We have considered all the gates to be made of 2 input AND, 2-input OR, and inverter (AOI). A 2-input XOR is composed of 2 AND, 1 OR, and 2 NOT gates. The area and delay of the 2-input AND, 2-input OR, and NOT gates (shown in Table I) are taken from the Synopsys Armenia Educational Department (SAED) 90-nm standard cell library datasheet for theoretical estimation. The area and delay of a design are calculated using the following relations:

$$A = a \cdot N_a + r \cdot N_o + i \cdot N_i \quad (5a)$$

$$T = n_a \cdot T_a + n_o \cdot T_o + n_i \cdot T_i \quad (5b)$$

where (N_a, N_o, N_i) and (n_a, n_o, n_i), respectively, represent the (AND, OR, NOT) gate counts of the total design and its critical path. (a, r, i) and (T_a, T_o, T_i), respectively, represent the area and delay of one (AND, OR, NOT) gate. We have calculated the (AOI) gate counts of each design for area and delay estimation. Using (5a) and (5b), the area and delay of each design are calculated from the AOI gate counts (N_a, N_o, N_i), (n_a, n_o, n_i), and the cell details of Table I.

TABLE I
AREA AND DELAY OF AND, OR, AND NOT GATES GIVEN IN THE SAED 90-nm S TANDARD CELL LIBRARY DATASHEET

	AND-gate	OR-gate	NOT-gate
Area (um^2)	7.37	7.37	6.45
Delay (ps)	180	170	100

B. Single-Stage CSLA

The general expression to calculate the AOI gate counts of the n-bit proposed CSLA and the BEC based CSLA of [6] and CBL-based CSLA of [7] and [8] are given in Table II of single stage design. We have calculated the AOI gate counts on the critical path of the proposed n-bit CSLA and CSLAs of [6]–[8] and used those AOI gate counts in (5b) to find an expression for delay of *final-sum* and *output-carry* in the unit of T_i (NOT gate delay).

For further analysis of the critical path of the proposed CSLA, the delay of each intermediate and output signals of the proposed n-bit CSLA design of Fig. 3 is shown in the square bracket against each signal. We can find from Table II that the proposed n-bit single-stage CSLA adder involves $6n$ less number of AOI gates than the CSLA of [6] and takes 2.7 and 6.6 units less delay to calculate *final-sum* and *output-carry*. Compared with the CBL-based CSLA of [7], the proposed CSLA design involves n more AOI gates, and it takes $(n - 4.7)$ unit less delay to calculate the *output-carry*.

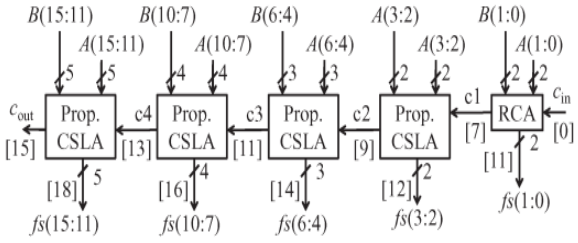


Fig. 3. Proposed SQRT-CSLA for $n = 16$. All intermediate and output signals are labeled with delay (shown in square brackets).

Simulation Results:

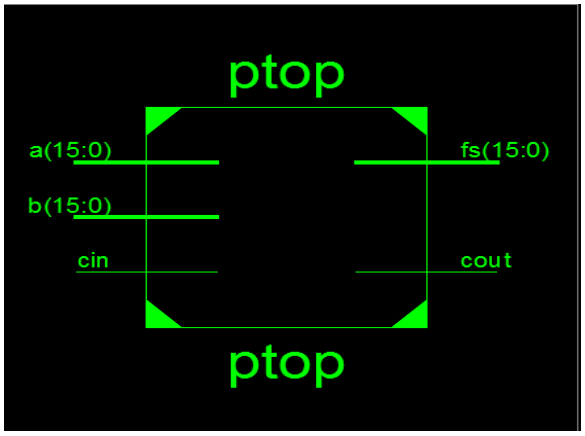


Fig 4:Block diagram

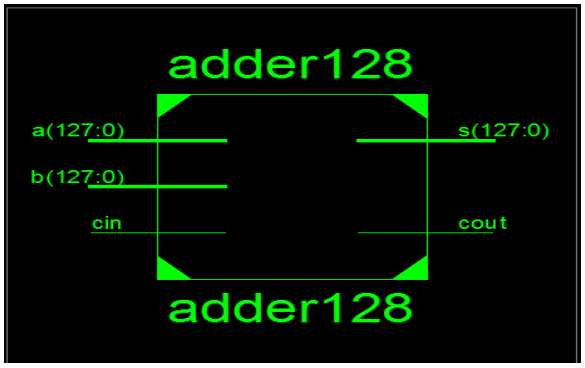


Fig 5.Block diagram of 128-bit adder

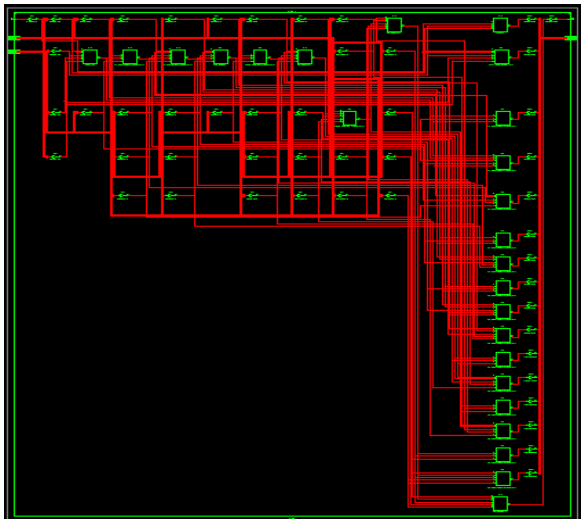


Fig 6:Technology schematic diagram.

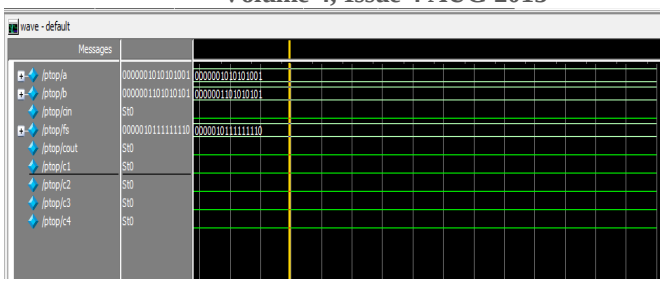


Fig7:Simulation results of proposed adder.

V.CONCLUSION

We have analyzed the logic operations involved in the conventional and BEC-based CSLAs to study the data dependence and to identify redundant logic operations. We have eliminated all the redundant logic operations of the conventional CSLA and proposed a new logic formulation for the CSLA. In the proposed scheme, the CS operation is scheduled before the calculation of *final-sum*, which is different from the conventional approach. Carry words corresponding to input-carry '0' and '1' generated by the CSLA based on the proposed scheme follow a specific bit pattern, which is used for logic optimization of the CS unit. Fixed input bits of the CG unit are also used for logic optimization. Based on this, an optimized design for CS and CG units are obtained. Using these optimized logic units, an efficient design is obtained for the CSLA. The proposed CSLA design involves significantly less area and delay than the recently proposed BEC-based CSLA. Due to the small carry output delay, the proposed CSLA design is a good candidate for the SQRT adder. The ASIC synthesis result shows that the existing BEC-based SQRT CSLA design involves 48% more ADP and consumes 50% more energy than the proposed SQRTCSLA, on average, for different bit-widths. In this project we are extended the bit width as 128-bits.

REFERENCES

[1] K. K. Parhi, *VLSI Digital Signal Processing*. New York, NY, USA: Wiley, 1998.
[2] A. P. Chandrakasan, N. Verma, and D. C. Daly, "Ultralow-power electronics for biomedical applications," *Annu. Rev. Biomed. Eng.*, vol. 10, pp. 247– 274, Aug. 2008.
[3] O. J. Bedrij, "Carry-select adder," *IRE Trans.*

Electron. Comput., vol. EC-11, no. 3, pp. 340–344, Jun. 1962.
[4] Y. Kim and L.-S. Kim, "64-bit carry-select adder with reduced area," *Electron. Lett.*, vol. 37, no. 10, pp. 614–615, May 2001.
[5] Y. He, C. H. Chang, and J. Gu, "An area-efficient 64-bit square root carryselect adder for low power application," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, vol. 4, pp. 4082–4085.
[6] 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.
[7] I.-C. Wey, C.-C. Ho, Y.-S. Lin, and C. C. Peng, "An area-efficient carry select adder design by sharing the common Boolean logic term," in *Proc. IMECS*, 2012, pp. 1–4.
[8] S. Manju and V. Sornagopal, "An efficient SQRT architecture of carry select adder design by common Boolean logic," in *Proc. VLSI ICEVENT*, 2013, pp. 1–5.
[9] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.