Area–Delay–Power Efficient Carry-Select Adder

Basant Kumar Mohanty, *Senior Member, IEEE*, and Sujit Kumar Patel

***Abstract*—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 *final-sum*, which is different from the conventional approach. Bit patterns of two anticipating carry words (corresponding to $c_{\rm in} = 0$ and 1) and fixed $c_{\rm in}$ 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, multistandard 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 ($c_{\rm in} = 0$ 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 SQRT-CSLA 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.

Based on the proposed logic formulation, we have derived an efficient logic design for CSLA. Due to optimized logic units, the proposed CSLA involves significantly less ADP than the existing CSLAs. We have shown that the SQRT-CSLA using the proposed CSLA design involves nearly 32% less ADP and consumes 33% less energy than that of the corresponding SQRT-CSLA. The rest of this brief is organized as follows. Logic formulation of CSLA is presented in Section II. The proposed CSLA is presented in Section III and the performance comparison is presented in Section IV. The conclusion is given in Section V.

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.

Manuscript received December 8, 2013; accepted March 12, 2014. Date of publication April 23, 2014; date of current version June 13, 2014. This brief was recommended by Associate Editor M. M. Mansour.

The authors are with the Department of Electronics and Communication Engineering, Jaypee University of Engineering and Technology, Raghogarh 473 226, India (e-mail: bk.mohanti@juet.ac.in; sujit.patel@juet.ac.in).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2014.2319695

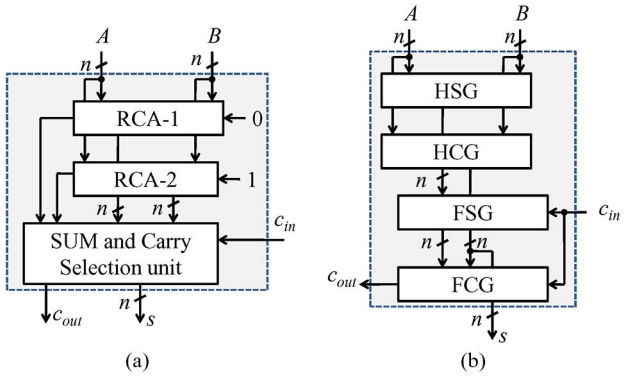

Fig. 1. (a) Conventional CSLA; n is the input operand bit-width. (b) The logic operations of the RCA is shown in split form, where HSG, HCG, FSG, and FCG represent half-sum generation, half-carry generation, full-sum generation, and full-carry generation, respectively.

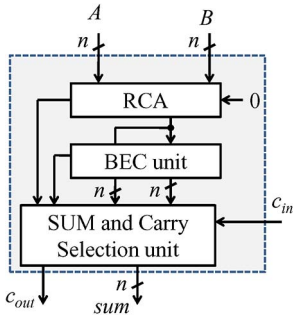

Fig. 2. Structure of the BEC-based CSLA; n is the input operand bit-width.

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) *full-carry* 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^0_{out} and c^1_{out}) corresponding to input-carry ($c_{\text{in}} = 0$ and $c_{\text{in}} = 1$), respectively. Logic expressions of RCA-1 and RCA-2 of the SCG unit of the n-bit CSLA are given as

$$s^0_0(i) = A(i) \oplus B(i) \quad c^0_0(i) = A(i) \cdot B(i) \tag{1a}$$
$$s^0_1(i) = s^0_0(i) \oplus c^0_1(i-1) \tag{1b}$$
$$c^0_1(i) = c^0_0(i) + s^0_0(i) \cdot c^0_1(i-1) \quad c^0_{\text{out}} = c^0_1(n-1) \tag{1c}$$
$$s^1_0(i) = A(i) \oplus B(i) \quad c^1_0(i) = A(i) \cdot B(i) \tag{2a}$$
$$s^1_1(i) = s^1_0(i) \oplus c^1_1(i-1) \tag{2b}$$
$$c^1_1(i) = c^1_0(i) + s^1_0(i) \cdot c^1_1(i-1) \quad c^1_{\text{out}} = c^1_1(n-1) \tag{2c}$$

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 $\{s^0_0(i), c^0_0(i)\}$ is identical to that of $\{s^1_0(i), c^1_0(i)\}$. 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-Based CSLA

As shown in Fig. 2, the RCA calculates n-bit *sum* s^0_1 and c^0_{out} corresponding to $c_{\text{in}} = 0$. 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

$$s^1_1(0) = \overline{s^0_1(0)} \quad c^1_1(0) = s^0_1(0) \tag{3a}$$
$$s^1_1(i) = s^0_1(i) \oplus c^1_1(i-1) \tag{3b}$$
$$c^1_1(i) = s^0_1(i) \cdot c^1_1(i-1) \tag{3c}$$
$$c^1_{\text{out}} = c^0_1(n-1) \oplus c^1_1(n-1) \tag{3d}$$

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_1 depends on s^0_1, which otherwise has no dependence on s^0_1 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.

It is interesting to note from (1a)–(1c) and (2a)–(2c) that logic expressions of s^0_1 and s^1_1 are identical except the terms c^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 $\{s_0, c_0, c_{\text{in}}\}$, where $c_0 = c^0_0 = c^1_0$. 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^0_1, s^1_1) for the *final-sum* of the CSLA. We find that a significant amount of logic resource is spent for calculating $\{s^0_1, s^1_1\}$, 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^0_1 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

$$s_0(i) = A(i) \oplus B(i) \quad c_0(i) = A(i) \cdot B(i) \tag{4a}$$
$$c^0_1(i) = c^0_1(i-1) \cdot s_0(i) + c_0(i) \quad \text{for } \left(c^0_1(0) = 0\right) \tag{4b}$$
$$c^1_1(i) = c^1_1(i-1) \cdot s_0(i) + c_0(i) \quad \text{for } \left(c^1_1(0) = 1\right) \tag{4c}$$
$$c(i) = c^0_1(i) \quad \text{if } (c_{\text{in}} = 0) \tag{4d}$$
$$c(i) = c^1_1(i) \quad \text{if } (c_{\text{in}} = 1) \tag{4e}$$

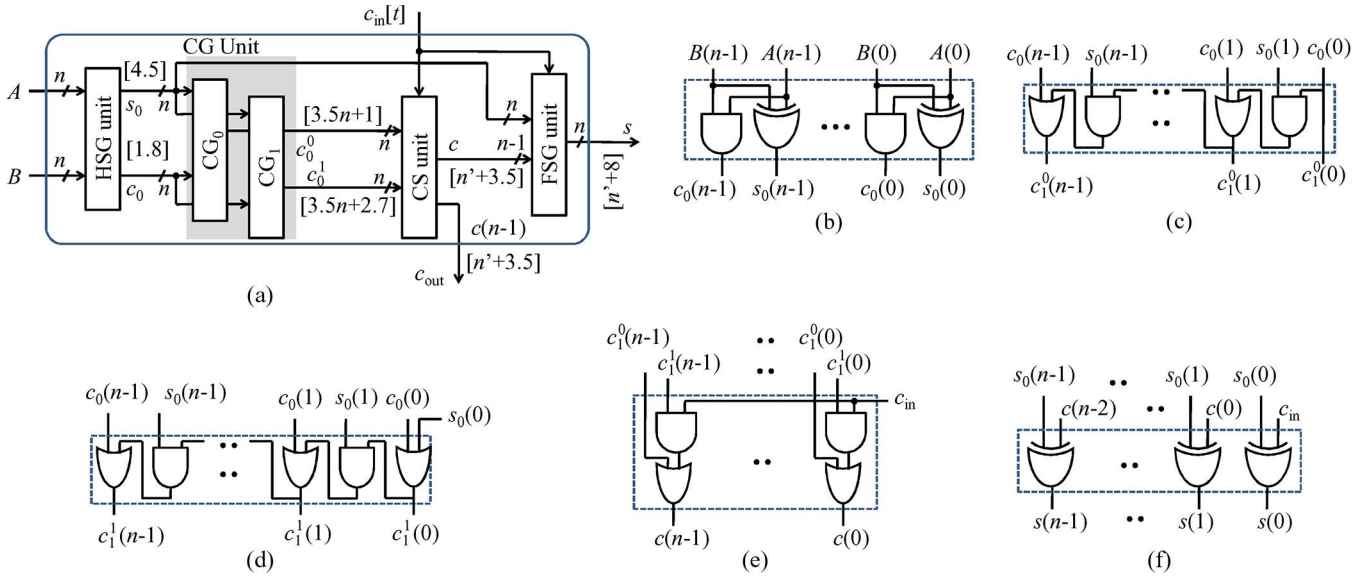

Fig. 3. (a) Proposed CS adder design, where n is the input operand bit-width, and [*] represents delay (in the unit of inverter delay), $n = \max(t, 3.5n + 2.7)$. (b) Gate-level design of the HSG. (c) Gate-level optimized design of (CG_0) for input-carry $= 0$. (d) Gate-level optimized design of (CG_1) for input-carry $= 1$. (e) Gate-level design of the CS unit. (f) Gate-level design of the final-sum generation (FSG) unit.

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

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

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_1^0 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_1^0 when $c_{\text{in}} = 0$; otherwise, it selects c_1^1. The CS unit can be implemented using an n-bit 2-to-1 MUX. However, we find from the truth table of the CS unit that carry words c_1^0 and c_1^1 follow a specific bit pattern. If $c_1^0(i) =$ '1', then $c_1^1(i) = 1$, irrespective of $s_0(i)$ and $c_0(i)$, for $0 \le i \le n-1$. 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.

IV. 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:

TABLE I
Area and Delay of AND, OR, and NOT Gates Given in the SAED 90-nm Standard Cell Library Datasheet

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

$$A = a \cdot N_a + r \cdot N_o + i \cdot N_i \tag{5a}$$

$$T = n_a \cdot T_a + n_o \cdot T_o + n_i \cdot T_i \tag{5b}$$

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.

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). The delay of the n-bit single-stage CSLA is shown in Table II for comparison. For further analysis of the critical

TABLE II
GENERAL COMPARISON OF GATE COUNTS AND DELAY OF THE PROPOSED AND EXISTING CSLAS FOR SINGLE-STAGE DESIGN. n: INPUT BIT-WIDTH

Design	AND-gate (N_a)	OR-gate (N_o)	NOT-gate (N_i)	final-sum (T_{fs})	output-carry (T_{cout})
Conventional	$14n-4$	$7n-3$	$5n-1$	$\max(t, 3.5n+2)+4.5$	$\max(t, 3.5n+1)+4.5$
CSLA [6]	$11n-2$	$5n-1$	$7n$	$\max(t, 3.5n+8.3)+4.5$	$\max(t, 3.5n+8.3)+4.5$
CSLA [7]	$7n$	$4n$	$5n$	$4.5n+1.8$	$4.5n+1.8$
CSLA [8]	$14n-7$	$8n-4$	$9n-4$	$9n+1$	$9n+1$
Proposed	$8n-2$	$5n-1$	$4n$	$\max(t, 3.5n+2.7)+8$	$\max(t, 3.5n+2.7)+3.5$

t stands for delay of *input-carry*. $t=0$ for single stage adder design. Delay expressed in the unit of T_i (NOT-gate delay).

TABLE III
THEORETICAL ESTIMATE OF AREA AND DELAY COMPLEXITIES OF THE PROPOSED AND EXISTING CSLAS

Design	width (n)	Area (um^2)	Delay (ns)		ADP (um^2.us)	EADP (%)
			fs	c_{out}		
CONV	8	1438.12	3.45	3.35	4.96	35
	16	2934.28	6.25	6.15	18.34	43
CSLA [6]	8	1282.45	4.08	4.08	5.23	42
	16	2587.01	6.88	6.88	17.80	39
CSLA [7]	8	906.56	3.78	3.78	3.42	−7.0
	16	1813.12	7.38	7.38	13.38	4.0
CSLA [8]	8	1654.65	7.30	7.30	12.08	228
	16	3416.17	14.50	14.50	49.55	286
Prop.	8	951.09	3.87	3.42	3.68	−−
	16	1924.30	6.67	6.22	12.83	−−

CONV: conventional, ADP: area delay product, EADP: excess ADP over the proposed design, ADP: area× delay, delay of fs represents adder delay.

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*.

Using the expressions of Table II and AOI gate details of Table I, we have estimated the area and delay complexities of the proposed CSLA and the existing CSLA of [6]–[8], including the conventional one for input bit-widths 8 and 16. For the single-stage CSLA, the input-carry delay is assumed to be $t=0$ and the delay of *final-sum* (f_s) represents the adder delay. The estimated values are listed in Table III for comparison. We can find from Table III that the proposed CSLA involves nearly 29% less area and 5% less output delay than that of [6]. Consequently, the CSLA of [6] involves 40% higher ADP than the proposed CSLA, on average, for different bit-widths. Compared with the CBL-based CSLA of [7], the proposed CSLA design has marginally less ADP. However, in the CBL-based CSLA, delay increases at a much higher rate than the proposed CSLA design for higher bit-widths. Compared with the conventional CSLA, the proposed CSLA involves 0.42 ns more delay, but it involves nearly 28% less ADP due to less area complexity. Interestingly, the proposed CSLA design offers multipath parallel carry propagation,

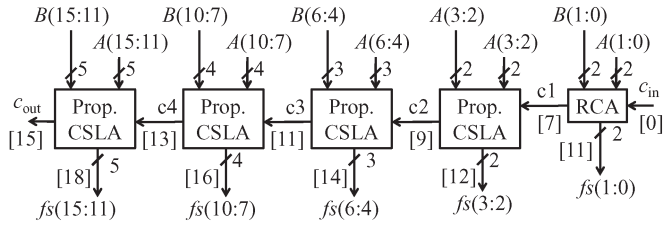

Fig. 4. Proposed SQRT-CSLA for $n=16$. All intermediate and output signals are labeled with delay (shown in square brackets).

whereas the CBL-based CSLA of [7] offers a single carry propagation path identical to the RCA design. Moreover, the proposed CSLA design has 0.45 ns less output-carry delay than the output-sum delay. This is mainly due to the CS unit that produces output-carry before the FSG calculates the *final-sum*.

C. Multistage CSLA (SQRT-CSLA)

The multipath carry propagation feature of the CSLA is fully exploited in the SQRT-CSLA [5], which is composed of a chain of CSLAs. CSLAs of increasing size are used in the SQRT-CSLA to extract the maximum concurrence in the carry propagation path. Using the SQRT-CSLA design, large-size adders are implemented with significantly less delay than a single-stage CSLA of same size. However, carry propagation delay between the CSLA stages of SQRT-CSLA is critical for the overall adder delay. Due to early generation of output-carry with multipath carry propagation feature, the proposed CSLA design is more favorable than the existing CSLA designs for area–delay efficient implementation of SQRT-CSLA. A 16-bit SQRT-CSLA design using the proposed CSLA is shown in Fig. 4, where the 2-bit RCA, 2-bit CSLA, 3-bit CSLA, 4-bit CSLA, and 5-bit CSLA are used. We have considered the cascaded configuration of (2-bit RCA and 2-, 3-, 4-, 6-, 7-, and 8-bit CSLAs) and (2-bit RCA and 2-, 3-, 4-, 6-, 7-, 8-, 9-, 11-, and 12-bit CSLAs), respectively, for the 32-bit SQRT-CSLA and the 64-bit SQRT-CSLA to optimize adder delay. To demonstrate the advantage of the proposed CSLA design in SQRT-CSLA, we have estimated the area and delay of SQRT-CSLA using the proposed CSLA design and the BEC-based CSLA of [6] and the CBL-based CSLA of [7] for bit-widths 16, 32, and 64 using Tables I, II, and (5a) and (5b). The estimated values are listed in Table IV for comparison. As shown in Table IV, the delay of the CBL-based SQRT-CSLA [7] is significantly higher for large bit-widths than the proposed SQRT-CSLA and BEC-based SQRT-CSLA designs. Compared with SQRT-CSLA designs of [6] and [7], the proposed SQRT-CSLA design, respectively, involves 35% and 72% less ADP, on average, for different bit-widths.

TABLE IV
THEORETICAL ESTIMATE OF AREA AND DELAY COMPLEXITIES OF THE PROPOSED AND EXISTING SQRT-CSLAS

Design	width (n)	Delay (ns)	Area (um^2)	ADP (um^2us)	EADP (%)
SQRT-CSLA	16	3.33	2287.41	7.62	48.76
(BEC)	32	4.28	4853.14	20.77	49.49
[6]	64	5.63	10006.73	43.77	49.39
SQRT-CSLA	16	7.38	1813.71	13.39	161.41
(CBL)	32	14.58	3627.42	52.89	280.64
[7]	64	28.98	7254.84	210.25	436.55
	16	3.0	1706.80	5.12	−−
Proposed	32	3.85	3608.98	13.89	−−
SQRT-CSLA	64	5.27	7435.46	39.18	−−

TABLE V
COMPARISON OF POSTLAYOUT APPLICATION-SPECIFIED INTEGRATED CIRCUIT (ASIC) SYNTHESIS RESULTS USING THE SAED 90-nm CMOS LIBRARY

Design	width (n)	DAT (ns)	Area (um^2)	Power (uW)
SQRT-CSLA	16	5.61	2890.52	30.5673
(CONV)	32	6.56	6100.34	60.2537
	64	8.37	12613.2	113.6452
SQRT-CSLA	16	5.96	2325.09	25.7958
(BEC)	32	7.64	4801.33	50.0750
[6]	64	10.18	9809.75	91.1774
SQRT-CSLA	16	10.45	1722.96	12.8662
(CBL)	32	18.72	2765.38	17.7900
[7]	64	35.10	5530.56	30.1427
	16	5.55	1813.45	19.6652
Proposed	32	6.59	3735.36	38.1886
SQRT-CSLA	64	8.35	7603.89	70.6244

D. ASIC Synthesis Results

We have coded the SQRT-CSLA in VHDL using the proposed CSLA design and the existing CSLA designs of [6] and [7] for bit-widths 16, 32, and 64. All the designs are synthesized in the Synopsys Design Compiler (DC) using the SAED 90-nm CMOS library. The netlist file obtained from the DC are processed in the IC Compiler (ICC). After placement and route, the area, data-arrival time (DAT), and power reported by the ICC are listed in Table V for comparison. The synthesis result of Table V confirms the theoretical estimates given in Table IV. As shown in Table V, the proposed SQRT-CSLA involves significantly less area and less delay and consumes less power than the existing designs. We can find from Fig. 5 that the proposed SQRT-CSLA design offers a saving of 39% ADP and 37% energy than the RCA-based conventional SQRT-CSLA; 32% ADP and 33% energy than the BEC-based SQRT-CSLA of [6]; and 55% ADP and 30% energy than the CBL-based SQRT-CSLA of [7], on average, for different bit-widths.

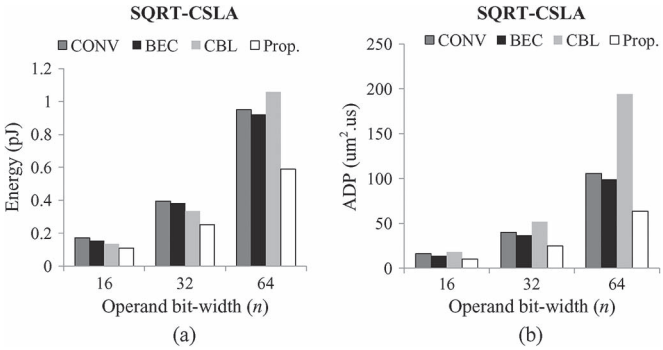

Fig. 5. (a) Comparison of energy consumption. (b) Comparison of ADP.

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 SQRT-CSLA, on average, for different bit-widths.

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 carry-select 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.