ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

An Improved DELAY-POWER Efficient Modified Carry Select Adder Design & Implementation Using VHDL

Ravi Kiran Kurella
Research Scholar-PG,
Department Of ECE,
Swarnandhra College of Engineering & Techonology, Seetharampuram,
Narsapur, W.G.Dist,
Andhra Pradesh, India.

J.E.N Abilash
Associate Professor,
Department Of ECE,
Swarnandhra College of Engineering & Techonology, Seetharampuram,
Narsapur, W.G.Dist,
Andhra Pradesh, India.

N.Srikanth
Associate Professor,
Department Of ECE,
Swarnandhra College of Engineering & Techonology, Seetharampuram,
Narsapur, W.G.Dist,
Andhra Pradesh, India.

Abstract:

In the present world all digital applications requires many arithmetic operations like addition and multiplications. There are several techniques for implementing addition operations like ripple carry adder,carry look ahead adder,carry skip adder,carry save adder and carry select adder. Our paper mainly described the brief 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 cin = 0 and 1) and fixed cin 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 power-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 power–delay–product (PDP) 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 PDP 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 [2], [3]. 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 (cin = 0 and 1) and selects one out of each pair for final-sum and final-output-carry [4]. 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 [5] 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. [6] 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Ramkumar and Kittur [7] 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 [8] and [9]. The CBL-based CSLA of [8] 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 [9]. However, the CBL-based SQRTCSLA design of [9] 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 extensively contributes to the critical path. Different logic designs have been suggested for efficient execution of the SCG unit. We made a study of the logic designs recommended for the SCG unit of conventional and BEC-based CSLAs of [7] by proper 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.

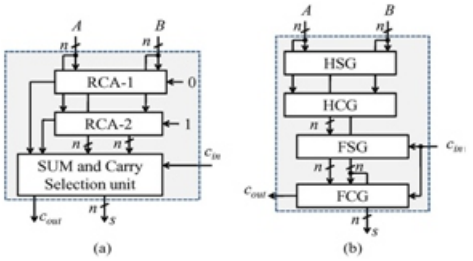


Fig. 1. (a) Conventional CSLA; n is the input operand bit-width. (b) The logic operations of the RCA is shown in split form, where HSG, HCG, FSG, and FCG represent half-sum generation, half-carry generation, full-sum generation, and full-carry generation, respectively.

A. SCG Unit of the Conventional CSLA:

As shown in Fig. 1(a), the SCG unit of the conventional CSLA [4] 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 (s0 and s1) and output-carry (c0 out and c1 out) corresponding to input-carry (cin = 0 and cin = 1), respectively Logic expressions of RCA-1 and RCA-2 of the SCG unit of the n-bit CSLA are given as

$$s_0^0(i) = A(i) \oplus B(i) \quad c_0^0(i) = A(i) \cdot B(i) \tag{1a}$$

$$s_1^0(i) = s_0^0(i) \oplus c_1^0(i-1) \tag{1b}$$

$$c_1^0(i) = c_0^0(i) + s_0^0(i) \cdot c_1^0(i-1) \quad c_{out}^0 = c_1^0(n-1) \tag{1c}$$

$$s_0^1(i) = A(i) \oplus B(i) \quad c_0^1(i) = A(i) \cdot B(i) \tag{2a}$$

$$s_1^1(i) = s_0^1(i) \oplus c_1^1(i-1) \tag{2b}$$

$$c_1^1(i) = c_0^1(i) + s_0^1(i) \cdot c_1^1(i-1) \quad c_{out}^1 = c_1^1(n-1) \tag{2c}$$

where c01 (–1) = 0, c11 (–1) = 1, and 0 ≤ i ≤ n – 1.

The logic expression of {s00 (i), c00 (i)} is identical to that of {s10 (i), c10 (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, [5] and [6] have used an add-one circuit instead of RCA-2 in the CSLA, in which a BEC circuit is used in [7] for the same purpose.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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. SCG Unit of the BEC-Based CSLA:

As shown in Fig. 2, the RCA calculates n-bit sum s01 and c0 out corresponding to cin = 0. The BEC unit receives s01 and c0 out from the RCA and generates (n + 1)-bit excess-1 code. The most significant bit (MSB) of BEC represents c1 out, in which n least significant bits (LSBs) represent s11 . The logic expressions.

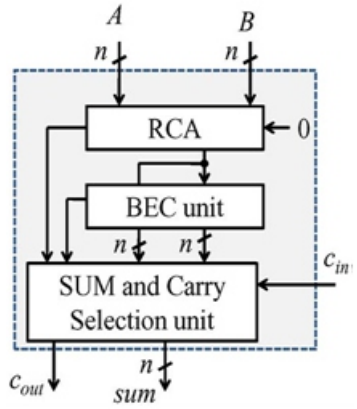


Fig. 2. Structure of the BEC-based CSLA; n is the input operand bit-width

of the RCA are the same as those given in (1a). The logic expressions of the BEC unit of the n-bit BEC-based CSLA are given as

$$s_1^1(0) = \overline{s_1^0}(0) \quad c_1^1(0) = s_1^0(0) \tag{3a}$$
$$s_1^1(i) = s_1^0(i) \oplus c_1^1(i-1) \tag{3b}$$
$$c_1^1(i) = s_1^0(i) \cdot c_1^1(i-1) \tag{3c}$$
$$c_{\text{out}}^1 = c_1^0(n-1) \oplus c_1^1(n-1) \tag{3d}$$

for 1 ≤ i ≤ n – 1. We can find from (1a) in the case of the BEC-based CSLA, c11 depends on s01 , which otherwise has no dependence on s01 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 additional study on the data dependence to find an optimized logic expression for the CSLA. It is interesting to note from that logic expressions of s01 and s11 are identical except the terms c01 and c11since (s00 = s10 = s0). In addition, we find that c01 and c11depend on {s0, c0, cin}, where c0 = c00 = c10 .

Since c01 and c11 have no dependence on s01 and s11 , the logic operation of c01 and c11 can be scheduled before s01 and s11 , and the select unit can select one from the set (s01 , s11) for the final-sum of the CSLA. We find that a significant amount of logic resource is spent for calculating {s01 , s11 }, 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 {c0 and c1} to calculate the final-sum. The selected carry word is added with the half-sum (s0) to generate the final-sum (s). Using this method, one can have three design advantages:

1) Calculation of s01 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 unnecessary logic operations of.The proposed logic formulation for the CSLA is given as

$$s_0(i) = A(i) \oplus B(i) \quad c_0(i) = A(i) \cdot B(i) \tag{4a}$$
$$c_1^0(i) = c_1^0(i-1) \cdot s_0(i) + c_0(i) \quad \text{for } (c_1^0(0) = 0) \tag{4b}$$
$$c_1^1(i) = c_1^1(i-1) \cdot s_0(i) + c_0(i) \quad \text{for } (c_1^1(0) = 1) \tag{4c}$$
$$c(i) = c_1^0(i) \quad \text{if } (c_{\text{in}} = 0) \tag{4d}$$
$$c(i) = c_1^1(i) \quad \text{if } (c_{\text{in}} = 1) \tag{4e}$$

Existing Conventional CSLA:
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 [8] and [9] are given in the 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 to find an expression for delay of final-sum and output-carry in the unit of Ti (NOTgate delay). The delay of the n-bit single-stage CSLA is shown for comparison. For advance analysis of the criticalpath of the proposed CSLA, the delay of each transitional and output signals of the proposed n-bit CSLA design of The proposed n-bit single-stage CSLA adder involves 6n less number of AOI gates than the CSLA of [7] and takes less delay to calculate final-sum and output-carry. Compared with the CBL-based CSLA of [8], the proposed CSLA design involves n more AOI gates, and it takes (n – 4.7) unit less delay to calculate the output-carry.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Using the expressions of and AOI gate details of , 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 (fs) represents the adder delay. The probable values are listed .. We can find from that the proposed CSLA involves nearly 29% less area and 5% less output delay than that of [7]. Consequently, the CSLA of [7] involves 40% higher ADP than the proposed CSLA, on average, for different bit-widths. Compared with the CBL-based CSLA of [8], 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, th3256e proposed CSLA design offers multipath parallel carry propagation.

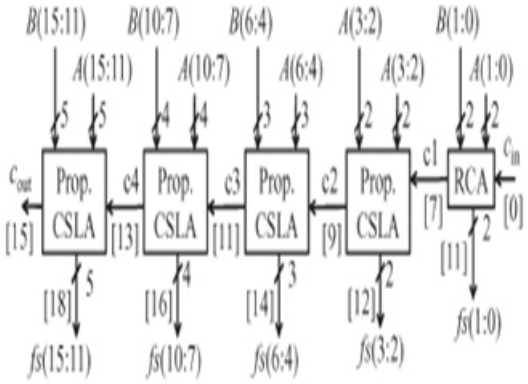


Fig. 3. SQRT-CSLA for n = 16. All intermediate and output signals are labeled with delay (shown in square brackets)

Where as the CBL-based CSLA of [8] 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 demoralized 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 implement with significantly less delay than a single-stage CSLA of same size. but, carry propagation delay between the CSLA stages of SQRT-CSLA is serious 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 SQRTCSLA and the 64-bit SQRT-CSLA to optimize adder delay.

To demonstrate the advantage of the proposed CSLA design inSQRT-CSLA, we have projected the area and delay of SQRTCSLA using the proposed CSLA design and the BEC-basedCSLA of [6] and the CBL-based CSLA of [7] for bit-widths 16, 32, and 64 are calculated. The estimated values are listed for comparison the delay of the CBL-based SQRT-CSLA [8] is significantly higher for large bit-widths than the proposedSQRT-CSLA and BEC-based SQRT-CSLA designs. Compared with SQRT-CSLA designs of [7] and [8], the proposed SQRTCSLA design, respectively, involves.

III. PROPOSED ADDER DESIGN:

The proposed CSLA is based on the logic formulation given in (4a), 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 (CG0 and CG1) corresponding to input-carry '0' and '1'. The HSG receives two n-bit operands (A and B) and generate half-sum word s0 and half-carry word c0 of width n bits each. Both CG0 and CG1 receive s0 and c0 from the HSG unit and generate two n-bit full-carry words c01 and c11 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 CG0 and CG1 are optimized to take advantage of the fixed input-carry bits. The optimized designs of CG0 and CG1 are shown in Fig. 3(c) and (d), correspondingly

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

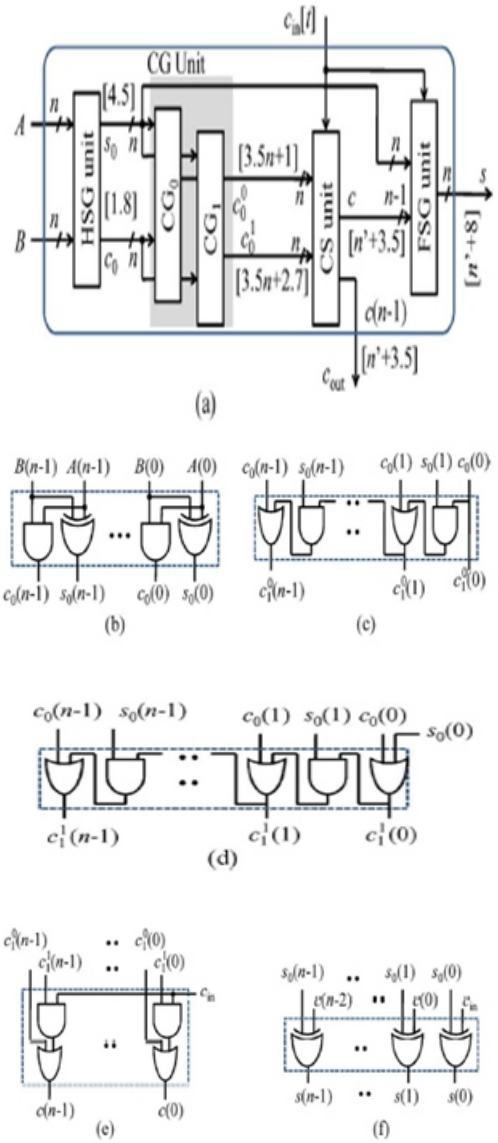


Fig. 4. (a) Proposed CS adder design, where n is the input operand bit-width, and [] represents delay (in the unit of inverter delay), n = max(t, 3.5n + 2.7). (b) Gate-level design of the HSG. (c) Gate-level optimized design of (CG0) for input-carry = 0. (d) Gate-level optimized design of (CG1) for input-carry = 1 (e) Gate-level design of the CS unit. (f) Gate-level design of the final-sum generation (FSG) unit.

$$c_{out} = c(n-1)$$
$$s(0) = s_0(0) \oplus c_{in} \quad s(i) = s_0(i) \oplus c(i-1).$$

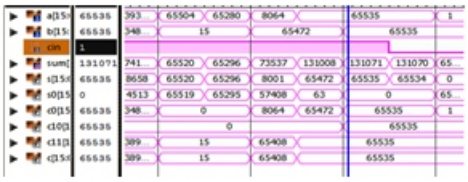
The CS unit selects one final carry word from the two carry words available at its input line using the control signal cin. It selects co1 when cin = 0; otherwise, it selects c11. The CS unit can be implemented using an n-bit 2-to-l MUX.

However, we find from the truth table of the CS unit that carry words co1 and c11 follow a specific bit pattern. If co1 (i) = '1', then c11 i) = 1, irrespective of so(i) and co(i), for 0 ≤ i ≤ 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 collected of n AND–OR gates. The final carry word c is obtain from the CS unit. The MSB of c is sent to output as cout, and (n – 1) LSBs are XORed with (n – 1) MSBs of half-sum (so) in the FSG [shown in Fig. 3(f)] to obtain (n – 1) MSBs of final-sum (s). The LSB of so is XORed with cin to obtain the LSB of s.

Modified 8-bit CSLA:

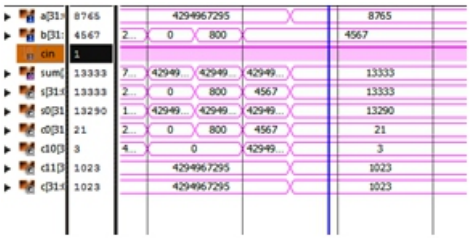
Simulation wave forms for Modified 8-bit CSLA

Modified 16-bit CSLA



Simulation wave forms for Modified 16-bit CSLA

Modified 32-bit CSLA



Simulation wave forms for Modified 32-bit CSLA

PERFORMANCE COMPARISON:

The Performance Comparison for the Modified CSLA is tabulated, which is designed by using HSG, CS, FSG Units and CG Unit(for i/p – Carry = 0 & 1).The under mentioned table shows Delay, Power and PDP Calculated and comparing with different adders of varied bits. The Modified CSLA Proven with better results Compared with Conventional CSLA.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Adder Type	No of bits	Delay	No Of Slices	No Of LUT' S	Power (uW)	PDP
RCA	8	16.976ns	15	21	18.546	314.83
RCA	16	29.986ns	28	43	28.253	847.19
RCA	32	37.723ns	36	65	56.565	2133.80
CSLA	8	20.243ns	9	18	15.362	310.97
CSLA	16	25.824ns	24	69	29.563	763.43
CSLA	32	28.561ns	54	101	68.938	1968.93
BEC CSLA	8	20.631ns	5	15	14.346	295.972
BEC CSLA	16	26.174ns	26	58	24.894	651.575
BEC CSLA	32	29.168ns	51	91	63.842	1862.14
MOD CSLA	8	18.717ns	18	31	15.456	289.28
MOD CSLA	16	21.55ns	37	65	28.976	624.432
MOD CSLA	32	23.126ns	77	142	65.983	1525.92

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 C 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 power-delay product 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 PDP and consumes 50% more energy than the proposed SQRTCSLA, on average, for different bit-widths like 8,16 and 32 bits.

REFERENCES:

[1] Basant Kumar Mohanty and Sujith Kumar Patel, " Area – Delay – Power Efficient Carry-Select Adder,"IEEE Transactions On Circuits and Systems – II : Express Briefs, vol. 61, No. 6, pp. 418–422, June 2014.

[2] K. K. Parhi, VLSI Digital Signal Processing. New York, NY, USA:Wiley,1998.

[3] 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.

[4] O. J. Bedrij, "Carry-select adder," IRE Trans. Electron. Comput., vol. EC-11, no. 3, pp. 340–344, Jun. 1962.
[5] 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.

[6] Y. He, C. H. Chang, and J. Gu, "An area-efficient 64-bit square root carryselectadder for low power application," in Proc. IEEE Int. Symp. Circuits Syst., 2005, vol. 4, pp. 4082–4085.

[7] 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.

[8] 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.

[9] S.Manju and V. Sornagopal, "An efficient SQRT architecture of carry selectadder design by common Boolean logic," in Proc. VLSI ICEVENT, 2013, pp. 1–5.

[10] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.