International Journal For Technological Research In Engineering
Volume 4, Issue 7, March-2017 *ISSN (Online): 2347 - 4718*

EFFICIENT SQUARE ROOT CARRY SELECT ADDER

Nagaraj Guler[1], Vijaya Madhavi[2]
[2]Associate Professor, [1,2]Department of ECE, EPCET, Bengaluru, Karnataka

Abstract: In this paper the logic operations involved in binary to excess-1 converter (BEC) based Carry select adder and regular Carry select adder (CSLA) are analyzed to identify redundant logic operations. Eliminating all redundant logic operations present in the conventional CSLA, a new logic formulation for CSLA is proposed. In the proposed technique, the carry select (CS) operation is scheduled before the calculation of final sum as output, which is different from the conventional approach. Fixed Cin (0 and 1) bits are used for logic optimization of carry select and generation units. The proposed CSLA design involves less area, delay and power than the conventional and BEC based CSLA. In this proposed 16-bit design area, delay and power is reduced by 37%, 9.9% and 35.6% respectively compared to 16-bit SQRT Regular CSLA and by 22%, 17.7%, 23.7% respectively when compared to 16-bit BEC based CSLA.

Keywords: Carry select adder, Ripple carry adder, Binary to excess-1 conversion, Sum carry generation, Sum carry selection.

I. INTRODUCTION

Area efficient, high performance and low power VLSI systems are increasingly used in portable, mobile, biomedical devices. An adder is the main part of an arithmetic unit. A complex Digital signal processor (DSP) involves several adders. An efficient adder design essentially improves the performance and computation speed of a complex DSP system. A ripple carry adder (RCA) uses a simple design but carry propagation delay is the main concern in this adder. Carry look-ahead adder and carry select (CS) methods have been suggested to reduce the carry propagation delay of adders of higher bit length which intern increases the efficiency of it [1]. We observe that logic optimization mainly depends on redundant operations available in the formulation, whereas adder delay largely depends on data dependence. In these existing designs, logic is optimized without giving any consideration to the data dependence. In this brief, we approach an analysis on logic operations involved in both conventional and BEC-based CSLAs to study the data dependence and to identify redundant logic operations. Based on these analyses, we have proposed a logic formulation for the CSLA. The main contributions in this are due to logic formulation based on data dependence and optimized carry generator (CG) and CS design. Based on the proposed logic formulation, we have designed an efficient logic structure for CSLA. Due to optimized logic units, the proposed CSLA involves significantly less ADP than the existing CSLAs [2].

II. LITERATURE SURVEY

A conventional carry select adder (CSLA) is RCA-RCA configurations that generates a pair of sum and carry output bits Corresponding to the anticipated input carry cin (0 and 1) and selects one out of each pair for final sum and carry output [3]. It has less carry propagation delay (CPD) than an RCA, but the design is not much efficient since it uses a dual RCA structure. Kim and Kim [4] used one RCA and an add-one circuit replacing the two RCAs, wherein the add-one circuits design is implemented using a multiplexer (MUX). He et al. [5] proposed a square-root (SQRT)-CSLA with less delay to implement large bit-width adders. In a SQRT-CSLA, CSLAs with increasing bit width size are connected in a cascading structure. The SQRT-CSLA design provides a parallel path for carry propagation that reduces the overall adder delay. Ram kumar and Kittur [6] suggested a binary to BEC-based CSLA. The BEC-based CSLA involves less logic resources as compared to the conventional CSLA, but it has marginally higher delay. A CSLA based on common Boolean logic (CBL) is proposed in [7] and [8]. The CBL-based CSLA of [7] significantly utilizes less logic resource than that of conventional CSLA but it has longer CPD, which is almost equal to that of the RCA. To overcome this problem, SQRT-CSLA based on CBL was proposed [8]. However, the CBL-based SQRT-CSLA design of [8] utilizes more logic resource and more delay than the BEC-based SQRT-CSLA of [6].

III. LOGIC FORMULATION

The CSLA consists of two units: 1) the *sum* and *carry* generator unit (SCG) and 2) the *sum* and *carry* selection unit. The SCG unit utilizes most of the logic resources of CSLA and significantly contributes to the critical path. There are many different logic designs proposed for efficient implementation of the SCG unit. We made a brief study of such 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 used and required data dependency. Accordingly, we remove all possible redundant logic operations [6].

A. SCG Unit Logic Expressions of the Conventional CSLA

The logic operations of the RCA are shown in split form, where HSG represent half-sum generation, HCG - half-carry generation, FSG - full-sum generation, and FCG - full-carry generation, respectively. 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 bit-width of adder.

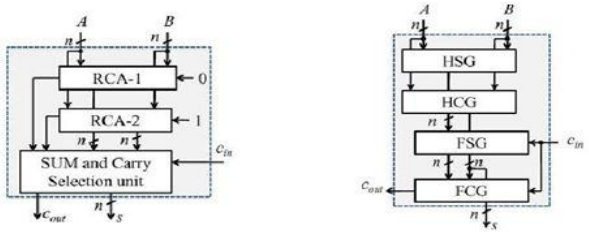


Fig.1: (a) Conventional CSLA (b) The logic operations of the RCA are shown in split form.

The logic operation of the *n*-bit RCA is performed in four stages in the given sequence: 1) *half-sum* generation (HSG) 2) *half-carry* generation (HCG) 3) *full-sum* generation (FSG) and 4) *full carry* generation (FCG). When two *n*-bit operands are added in the conventional CSLA, RCA-1 and RCA-2 generate *n*-bit *sum* (s_0 and s_1) and output carry corresponding to input-carry c_{in} = 0 and 1, respectively. Logic expressions of RCA-1 and RCA-2 of the Sum and Carry generator (SCG) unit of the *n*-bit CSLA are given as,

$s_0^{o}(i)=A(i)\ \ B(i)$ $\quad c_o^{o}(i)=A(i)\ B(i)$ (1a)

$s_1^{0}(i)= s_o^{o}(i)\ c_1^{o}(i-1)$ $\quad c_1^{0}=c_o^{o}(i)+s_o^{o}(i).c_1^{o}(i-1)$ (1b)

$c_{out}^{0}=c_1^{0}(n-1)$ (1c)

$s_0^{1}(i)=A(i)\ \ B(i)$ $\quad c_o^{1}(i)=A(i)\ B(i)$ (2a)

$s_1^{1}(i)= s_o^{1}(i)\ c_1^{1}(i-1)$ $\quad c_1^{1}(i)=c_o^{1}(i)+s_o^{1}(i)\ .\ c_1^{1}(i-1)$ (2b)

$c_{out}^{1}=c_1^{1}(n-1)$ (2c)

where $0 \le i \le n-1$.

As shown in eq(1a)–(1c) and (2a)–(2c), the logic expression of $\{s_0^0(i), c_0^0(i)\}$ similar to $\{s_0^1(i), c_0^1(i)\}$.These redundant logic operations can be removed to have an optimized design for CSLA, in which the HSG and HCG of RCA-1 is shared to construct RCA-2. Based on this, used an add-one circuit which is BEC circuit instead of RCA-2 in the CSLA. Since the results shows that BEC-based CSLA offers the best area, delay, power efficiency among the existing CSLAs.

B. SCG Unit Logic Expression of the BEC-Based CSLA

As shown in Fig 2, the RCA calculates *n*-bit *sum* s_1^0 and c_{out}^0 corresponding to c_{in} = 0. The BEC unit receives s_1^0 and c_{out}^0 from the RCA and generates (*n* + 1) bit excess-1 code. The most significant bit (MSB) of BEC represents c_{out}^1, 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).

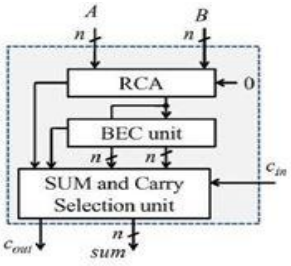


Fig 2: Structure of the BEC-based CSLA

$s_1^1(0)= \sim s_1^0(i)\ c_1^1(0)= s_1^0(0)$ (3a)

$s_1^1(i)=s_1^0(i)\ c_1^1(i-1)$ (3b)

$c_1^1(i)=s_1^0(i)\ \ c_1^{\ 1}(i-1)$ (3c)

$c_{out}^1=c_1^0(n-1)\ c_1^1(n-1)$ (3d)

for $1 \le i \le n-1$.

We can find from eq (1a)–(1c) and (3a)–(3d) that, in case of the BEC-based CSLA,c_1^1 depends on s_1^0 which otherwise has no dependence on s_1^0 in case of the conventional CSLA. Therefore the BEC method increases data dependence in CSLA. We have considered logic expressions of the conventional CSLA and made further study on the data dependence to find an optimized logic expression for the CSLA. It is interesting to know from (1a)–(1c) and (2a)–(2c) that logic expressions of s_1^0 and s_1^1 are identical except the terms c_1^0 and c_1^1 since ($s_0^0=s_0^1=s_0$).In addition, we find that c_1^0 and c_1^1 depend on {s_0, c_0, c_{in}}, where $c_0 = c_0^1 = c_0^0$. Since c_1^0 and c_1^1 have no reliance on s_1^0 and s_1^1, the logic operation of c_1^0 and c_1^1 can be reserved before s_1^0 and s_1^1, and the select unit can choose individual as of the set (s_1^0, s_1^1) for the final sum of the CSLA. We find that significant amount of logic resource is spent for manipulating $\{s_1^0, s_1^1\}$. It is not an efficient approach to reject one sum-word after the computation. Instead, to figure the final sum individual can chose the significant carry word as of the expected carry words $\{c_0 \text{ and } c_1\}$.To generate the final sum (*s*),the chosen carry word is added with the half sum (s_0). Using this system, one can have three configuration points:

- Computation of s_1^0 is not done in the SCG unit;
- Rather than the (*n*+1) bit, the *n*-bit select unit is necessary;
- Small output-carry delay.

Every bit of these points ends up with an efficient circuit on behalf of the CSLA. We have eliminated every unnecessary operations of (1a)–(1c) with (2a)–(2c) and revamped expressions of (1a)–(1c) with (2a)–(2c) based on their reliance.The projected logic for the CSLA is specified as,

$s_o(i)=A(i)\ \ B(i)$ $\quad c_o(i)=A(i)\ B(i)$ (4a)

$c_1^0(i)=c_0(i)+s_o(i).c_1^{\ o}(i-1)$ for $c_1^0(0)=0$ (4b)

$c_1^1(i)=c_0(i)+s_o(i).c_1^{\ 1}(i-1)$ for $c_1^1(0)=1$ (4c)

$c(i)=c_1^0(i)$ if $c_{in}=0$ (4d)

$c(i)=c_1^1(i)$ if $c_{in}=1$ (4e)

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

$s(0)=s_0(0)\ \ c_i$

$s(i)=s_0(i)\ \ c(i-1)$ (4g)

IV. PROPOSED CSLA (ADDER) DESIGN

The proposed CSLA is based on the logic formulation given in eq (4a)–(4g), and its structure is shown in Fig. 3(a). It consists of one Half-sum generator (HSG) unit, one Full sum

generator (FSG) unit, one Carry generator (CG) unit, and one Carry selector (CS) unit. The CG unit is composed of two CGs that is 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 each *n* bits width. Both CG_0 and CG_1 receives s_0 and c_0 from the HSG unit and generate two individual *n*-bit full-carry words c_1^0 and c_1^1 simultaneously 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 generated two carry words, available at its input line control signal c_{in}. It selects c_1^0 when c_{in} = 0; otherwise, it selects c_1^1 . The CS unit can be implemented using an *n*-bit 2-to-l MUX. The design of the CS unit is shown in Fig. 3(e), which is composed of *n* AND–OR gates. The fin al 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*.

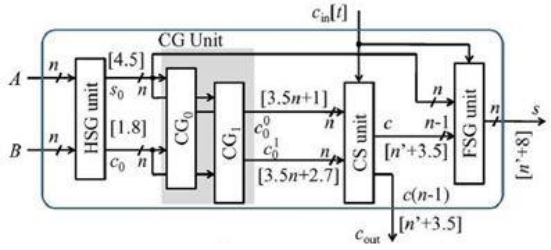
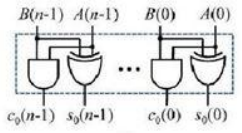
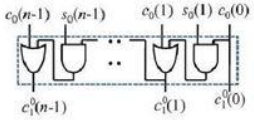


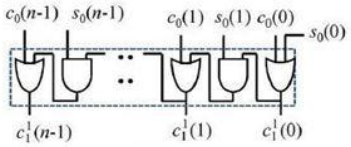
Fig 3: (a) Proposed CS adder design



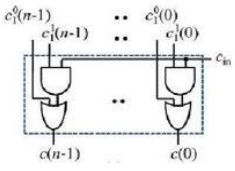
(b) HSG



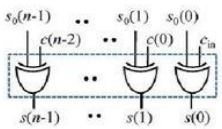
(c) CG0



(d) CG1



(e) CS unit



(f) FSG unit

Area–Delay Estimation

We have considered only 2-input AND, 2-input OR, and inverter (AOI) Logic. A 2-input XOR is composed of 2 AND gates, 1 OR gate, and 2 NOT gates. The area and delay of the 2-input AND gate, 2-input OR gate, and NOT gate (shown in Table I) are analyzed from the Cadence Tools using 180nm library.

TABLE 1: AREA, DELAY AND POWER OF AND, OR AND NOT GATES

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

The area and delay of a design are calculated using the following relations:

$$A=a \cdot N_a + r \cdot N_o + i \cdot N_i$$

$$T=n_a \cdot T_a + n_o \cdot T_o + n_i \cdot T_i$$

where (Na, No, Ni) and (na, no, ni) respectively, represent the (AND, OR, NOT) gate counts of the total design and also its critical path. (a, r, i) and (Ta, To, Ti), represent the area and delay of one (AND, OR, NOT) gates respectively. We have calculated the (AOI) gate counts of each design for estimation of area and delay.

V. SIMULATION RESULTS

Regular, Modified and Proposed CSLA Simulation is carried out using Xilinx simulation tool and Spartan 3E as the target device. The waveforms are shown below.

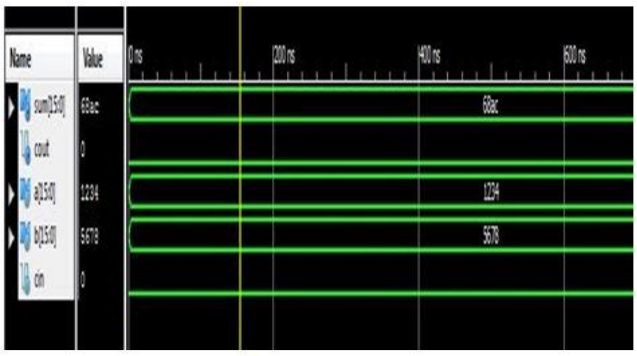


Fig 4.Simulation result for REGULAR CSLA.

Fig 4 is the output waveform for Regular CSLA, where 'a' and 'b' are inputs and c_{in} is input carry and got the output sum and carry with the same bits.

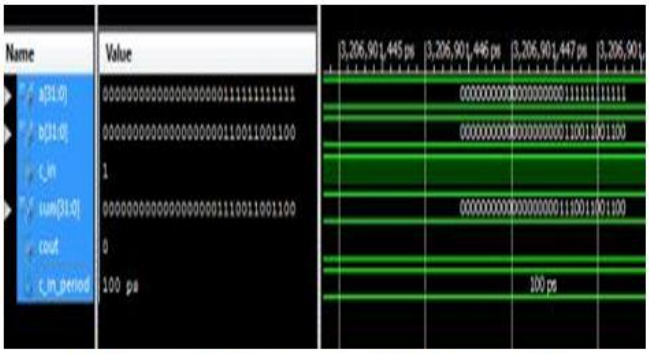


Fig 5. Simulation result for CSLA using BEC

Fig 5 is the output waveform for BEC based CSLA.Here c_in_period represents time period to pass a code from Ripple carry adder to BEC circuit.

Fig .6. Simulation result of proposed SQRT CSLA

Fig 6 is the output waveform for Proposed CSLA.Here carry_cout represents final carry.cou, cou1,cou2; cou3 represents individual carry of each full adders.

V. COMPARISON OF RESULTS

A comparative analysis of CSLA Adders with RCA, BEC and proposed in terms of area, delay and power is summarised in the table 2 and their respective graphs are shown in Fig 7(a), (b) and 7(c).

TABLE 2: ESTIMATION OF AREA ,DELAY AND POWER OF THE PROPOSED AND EXISTING SQRT CSLA

Adder width	Adder type	Area (um^2)	Delay (ns)	Power (uW)
4 bit	SQRT CSLA (REGULAR)	955.93	2.19	7.95
	SQRT CSLA (BEC)	628.90	3.33	6.68
	SQRT CSLA (PROPOSED)	434.84	2.01	5.72
8 bit	SQRT CSLA (REGULAR)	2016.09	3.94	15.29
	SQRT CSLA (BEC)	1362.03	4.84	13.25
	SQRT CSLA (PROPOSED)	952.34	3.23	10.41
16 bit	SQRT CSLA (REGULAR)	2890.52	5.61	30.56
	SQRT CSLA (BEC)	2325.09	6.14	25.79
	SQRT CSLA (PROPOSED)	1813.45	5.05	19.66

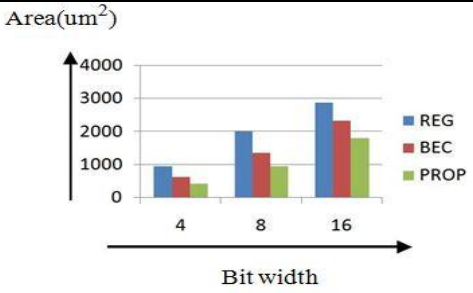


Fig: 7(a) Area

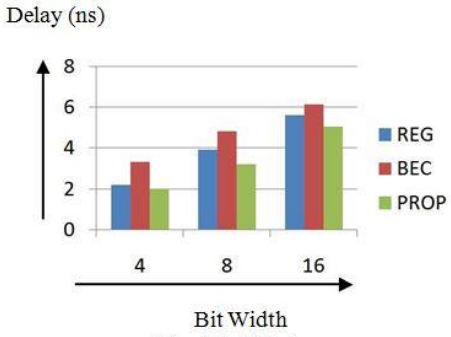


Fig 7(b) Delay

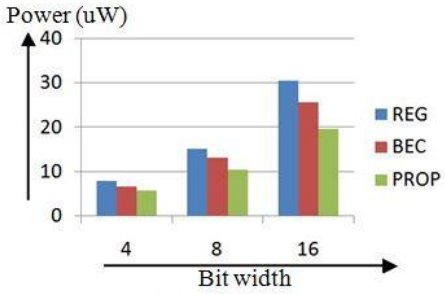


Fig 7(c) Power

It is observed that delay and number of logic levels for evaluation of sum and carry are still reduced when replaced with BEC based CSLA compared with Regular CSLA. For getting further optimized results BEC based CSLA is replaced with Proposed Carry Select Adder. The delay, number of logic levels for getting carry and sum and power utilized by the addition operation for 3 circuits are tabulated below. It is inferred that proposed CSLA gives better result. Analyzation of logic operations involved in the conventional (including BEC-based) CSLAs are studied, to identify data dependency and redundant logic operations. The redundant logic operations of the conventional CSLA are eliminated and a new logic formulation is proposed. In the proposed scheme, the Carry Select operation is scheduled before the calculation of final sum as output, which is used for logic optimization.

VI. FUTURE WORK

Present work gives the implementation by eliminating redundant logic operations in Regular CSLA. By using the BEC circuit instead of Carry generator 1 in proposed circuit gives further reduction in power and area consumption.

REFERENCES

[1] 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.

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

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

[10] M.Moris Mano, "Digital Design", Pearson Education, 3rd edition 2002.

[11] Singh, R.P.P.; Kumar, P.; Singh, B., "Performance Analysis of Fast Adders Using VHDL", Advances in Recent Technologies in Communication and Computing, 2009.

[12] A Tyagi. "A reduced area scheme for carry select adders", IEEE Trans. On computer, vol.42, pp.1163-1170,1993.

