International Organization of Research & Development (IORD)
ISSN: 2348-0831
Vol 05 Issue 01 | 2017

Delay & Area Efficient Carry-Select Adder

Gauravkumar D. Jade, Asst.Prof.Prafful Dubey, Prof.Vijay Sharma.
Lord Krishna College Of Technology, Indore (M.P).

ABSTRACT

In this paper, the logic operations occupied in conventional carry select adder (CSLA) and BEC- based carry select adder (CSLA) i.e., binary to excess-1 converter based carry select adder are estimated to study the data dependence and to identify redundant logic operations. We have removed all the redundant logic operations included in the conventional CSLA and proposed a new logic formation for CSLA. In the proposed idea, the carry select (CS) function is planned before the calculation of final-sum which is unlike from the conventional approach. Bit patterns of two expecting carry words (corresponding to cin = 0 and 1) and fixed cin bits are used for logic optimization of CS and generation units. A capable CSLA design is obtained using improved logic units. The proposed carry select adder (CSLA) design involves considerably less area and delay than the recently proposed BEC-based carry select adder. Due to the slight carry-output delay, the proposed CSLA design is a good nominee 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 finest between the existing SQRT-CSLA designs, for the different input bit- widths.

I. INTRODUCTION

Design of high speed digital adders with efficient area and delay is one of the significant areas of research in VLSI system design. Adders are the key components in general purpose microprocessors and digital signal processors systems. The RCA i.e., Ripple Carry Adders have the most compact design among the all kinds of adders, they are the slowest kinds of adders. While on the other hand, Carry Look-ahead Adders (CLAs) are the fastest adders, yet they are not so excellent from the area point of analysis. Carry Select Adders have been considered as a compromise solution between RCAs and CLAs because they offer a good tradeoff between the compact area of RCAs and the short delay of CLAs. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The addition for every bit position in an every adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position. The CSLA is used in several arithmetic systems to solve the problem of carry propagation delay by independently generating multiple carrys and then select a carry to generate the final sum [1] [2].

A conventional carry select adder (CSLA) is an RCA–RCA design that produces a pair of sum words and output carry bits matching the likely input-carry (cin = 0 and cin = 1) and selects one out of each pair for final-sum and final-output-carry [1]. A conventional CSLA has less carry propagation delay (CPD) than an RCA, but the design is not so fine since it uses a double RCA. Few efforts have been made to avoid dual use of RCA in CSLA design. Kim and Kim [4] used one RCA and one add-one circuit in its place of two RCAs, where the add-one circuit is implemented using a multiplexer (MUX). Y. He, C. H. Chang, and J. Gu. [7] 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 total adder delay. Here Ramkumar and Kittur [9] recommended a binary to BEC-based carry select adder (CSLA). The BEC-based carry select adder contains 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 [10] and [11]. The CBL-based CSLA of [10] 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 projected in [11]. Yet, the CBL-based SQRT-CSLA design of [11] requires more logic resource and delay than the BEC-based SQRT-CSLA of [9]. We observe that logic optimization largely depends on availability of redundant operations in the formulation, whereas adder delay mostly depends on data dependence. In the existing designs, logic is optimized without giving any consideration to the data dependence. In this paper, we prepared an analysis on logic operations occupied in conventional and BEC-based CSLAs to study the data dependence and to identify redundant logic operations. Based on this study, we have planned a new logic formulation for the CSLA. The major contribution in this paper is logic formulation based on data dependence and optimized carry generator (CG) and carry select unit (CS) design.

Based on the proposed logic formulation, we have found a capable logic design for CSLA. Due to better logic units, the projected 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 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 estimation method is presented in Section IV and 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 [12]. 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 [9] by suitable logic expressions. The main objective of this study is to identify redundant logic operations and data dependence. For that reason, we remove all redundant logic operations and sequence logic operations based on their data dependence. Fig. 1. (a) Conventional CSLA (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 [1] 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 and c1) corresponding to input-carry (cin = 0 and cin = 1), respectively. Logic out expressions of RCA-1 and RCA-2 of the SCG unit of the*n*-bit CSLA are given as

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

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

Where $c^0_1(-1) = 0$, $c^1_1(-1) = 1$, and $0 \le i \le n-1$.

The logic expression of $\{s^0(i), c^0(i)\}$ is matching to 0 that of $\{s^1(i), c^1(i)\}$. These redundant logic operations can be removed to have an optimized design for RCA-2, 0 0 in which the HSG and HCG of RCA-1is shared to create RCA-2. Depend on this, [4] and [7] have used an add- one circuit instead of RCA-2 in the CSLA, in which a BEC i.e., binary to excess one converter circuit is used in [9] for the same purpose. Since the BEC-based CSLA offers the best area–delay efficiency among the presented CSLAs, here we talk about 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 cin = 0. The BEC unit receives $^0_{out}$ and c^0 from the RCA and generates (n + 1)-bit excess-1 code. The most significant bit (MSB) of BEC represents c1 , in which n least significant bits (LSBs) represent s1 The logic expressions of the Ripple Carry out Adder (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

$s^1_1(0) = s^0_1(0)$ $c^1_1(0) = s^0_1(0)$ (3a)
$s^1_1(i) = s^0_1(i) \oplus c^1_1(i-1)$ (3b)
$c^1_1(i) = s^0_1(i) \cdot c^1_1(i-1)$ (3c)
$c^1_{out} = c^0_1(n-1) \oplus c^1_1(n-1)$ (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, c1 depends on s0 , which 1 1 otherwise has no dependence on s0 in the case of the conventional CSLA. The BEC method therefore increases data dependence in the carry select adder (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 exciting to note from (1a)–(1c) and (2a)–(2c) that logic expressions of s0and s^1 are identical except the terms $c^0$and c^1 since ($s^0 = s^1 = s$). In addition, we get that c^0 and c^1 depend on {s , c , cin}, where c = c0 1 1 0 0 0 1 1 0 0 0 0 = c1. Since c0 and c1 have no dependence on s0 and s1 , the logic operation of c0 and c1 can be scheduled

0 1 $_1$1 1 1 1 1 before s0 and s1 , and the select unit can select one from the set (s0 , s1) for the final-sum of the carry select 1 1 1 1 adder. We find that a major quantity of logic resource is spent for calculating {s0 , s1 }, and it is not an capable 1 1 approach to reject one sum-word after the calculation. In its place, 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 is avoided in the SCG unit. (2) The n-bit select unit is needed as a replacement for of the (n +bit. (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

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

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

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

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

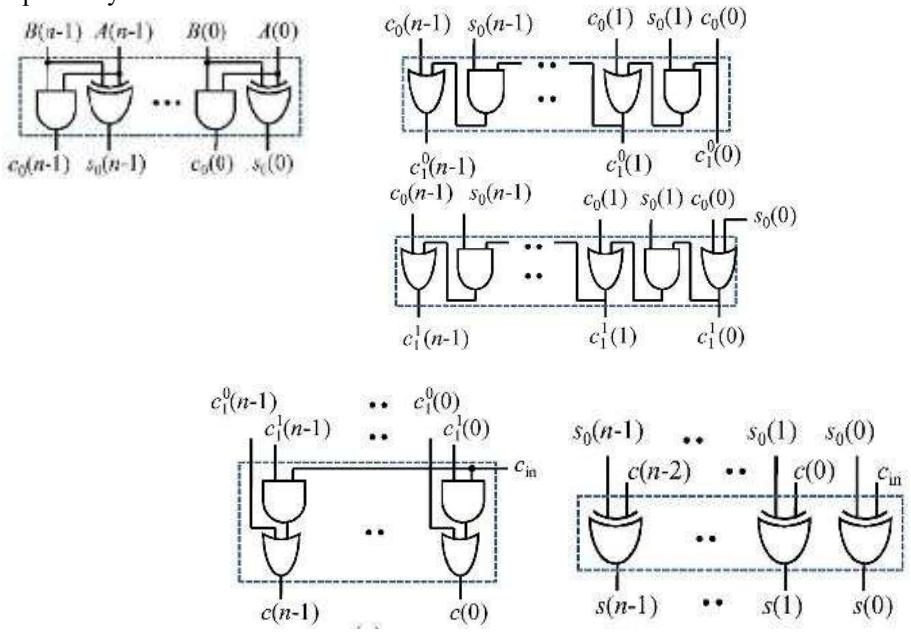
$c(i) = c^1{}_1(i)$ if $(cin = 1)$ (4e)

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

$s(0) = s_0(0) \oplus cin$ $s(i) = s_0(i) \oplus c(i-1)$. (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 s0 and half-carry word c0 of width n-bits each. Both CG0 and CG1 get s0 and c0from the HSG unit and generate two n-bit full-carry words c0 and c1 related to input-carry, Cin= 0 and Cin= 1, 11 correspondingly. The logic circuits of CG0 and CG1 are optimized to take advantage of the fixed input-carry bits.



Proposed CS adder design, where *n* is the input operand bit-width.(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.

The logic diagram of the HSG unit is ş̧hown 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. 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^0 and c^1 follow a definite bit pattern. If $c^0(i)$ = ¡1', at that time $c^1(i)$ =1, irrespectiv̧e of $s_0(i)$ and $c_0(i)$, for 0̧ ≤ 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. 5(e), which is composed of n-number of AND and OR gates. The final carry word c is gain from the CS unit. The most significant bit (MSB) of c is send 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 least significant bit (LSB) of s_0 is XORed with cin to obtain the LSB of s.

IV. PERFORMANC ESTIMATION METHOD

A. Area 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 made up of 2-AND, 1- OR, and 2-NOT gates. The area of the 2-input AND, 2-input OR, and a NOT gate (shown in Table 1) are taken from the Synopsys Armenia Educational Department (SAED) 90-nm standard cell library datasheet for area estimation.

Table.1. Area Of AND, OR, And NOT Gates Given In The SAED 90-Nm Standard Cell Library Datasheet.

AREA (μm^2)	AND-gate	OR-gate	NOT-gate
	7.37	7.37	6.45

In this project we estimate the total number of AOI gate used by the different design. This information is obtained from the advanced HDL synthesis report after the implementation of each design on Xilinx ISE design suit 12.2. The HDL synthesis report shows the how much number of 1-bit Adder, 2:1 Mux and XOR gates are used by each design. From this report we then calculate the total number of AOI gate utilized by each 1-bit Adder, 2:1 Mux and XOR gates in design. The area of a design is calculated using the following relations:

$$A = (a * Na) + (r * No) + (i * Ni)$$

Where (Na, No, Ni) signify the (AND, OR, NOT) gate counts of the entire design. Whereas (a, r, i) stand for the area of one (AND, OR, NOT) gate. We have calculated the (AOI) gate counts of each design for area estimation.

B. Delay Estimation Method

Here the delay for each design is estimated from the Post-PAR Static Timing Report. In this report we obtain the Data Sheet Report which shows the delay between the Source Pad and Destination Pad.

- For the delay estimation of 16-bit CSLA we have taken the delay between the Pad no (a< 15>) to (s<15>) which is final sum delay and (a <15>) to cout which is carry out delay.
- And for 32-b CSLA we have taken the delay between the Pad no (a< 31>) to (s< 31>) which is final sum delay and (a <31>) to cout which is carry out delay.

V. SYNTHESIS RESULTS

To demonstrate the advantage of the proposed CSLA design in SQRT-CSLA, we have shown the practical value of area and delay results of SQRT-CSLA using the proposed CSLA design, conventional CSLA design and the BEC-based CSLA design for bit-widths 16 and 32 using Xilinx ISE design suit as shown in Table 2. Here is the various synthesis results obtained after the implementation for each design.

Table 2. Practical Estimate of Area and Delay of the Proposed and Existing SQRT 16 and 32-bit CSLA.

Design	**Width (n)**	**Area (μm^2)**	**Delay (ns)**	**ADP (μm^2 . ns)**
			Fs	
CONV	16	2938.98	4.893	14.38
	32	5975.62	5.088	30.403
BEC	16	2338.29	5.769	13.48
	32	4696.85	5.650	26.537
PROP	16	1120.32	5.111	5.72
	32	2240.64	5.047	11.308

VI. CONCLUSION

We have estimated the logic operations occupied in the conventional and BEC-based CSLAs to study the data dependence and to identify redundant logic operations. We have remove all the unneeded logic operations of the conventional CSLA and proposed a new logic formulation for the CSLA. In the projected design, the carry select (CS) operation is scheduled before the calculation of final-sum, which is dissimilar 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 designs engage considerably less area and delay than the recently proposed BEC-based carry select adder. Due to the small carry output delay, the estimated CSLA design is an outstanding nominee for the SQRT adder. The FPGA synthesis result shows that the existing BEC-based SQRT-CSLA design involves more area delay product than the proposed SQRT-CSLA, for different bit-widths.

REFERENCE

[1]. O. J. Bedrij, "Carry-select adder," *IRE Trans. Electron. Comput.,* pp. 340–344, 1962.
[2]. J.Sklansky, "Conditional-Sum Addition Logic," *IRE Trans. Electron. Comput.,* vol. EC-9, pp.226-231, 1960.
[3]. T. Y. Ceiang and M. J. Hsiao, "Carry-select adder using single ripple carry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.
[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]. M.Alioto et.al, "A Gate Level Strategy To Design Carry Select Adders," *ISCAS 2004.*
[6]. Behnam Amelifard et.al "Closing the Gap between Carry Select Adder and Ripple Carry Adder," *Proceedings of the Sixth International Symposium on Quality Electronic Design (ISQED'05), 2005.*
[7]. 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.CircuitsSyst.*, 2005, vol. 4, pp. 4082–4085.
[8]. B. Ramkumar, H.M. Kittur, and P. M. Kannan, "ASIC Implementation Of Modified Faster Carry Save Adder," *Eur. J. Sci.Res.*, vol. 42, no. 1, pp. 53–58, 2010.
[9]. 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.
[10]. 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.
[11]. 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.
[12]. B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.