Design of Carry Select Adder for Low-Power and High Speed VLSI Applications

M. Vinod Kumar Naik,
Department of Electronics Engineering,
Pondicherry University,
Pondicherry, India.
vinodkr.pu@gmail.com

Mohammed Aneesh. Y,
Department of Electronics Engineering,
Pondicherry University,
Pondicherry, India.
aneeshssw1@gmail.com

***Abstract*-Low-power and high-performance VLSI systems are increasingly used in portable and mobile devices, multi-standard wireless receivers, and in biomedical applications. An adder is main component of an arithmetic unit. An efficient adder design essentially improves the performance of a complex DSP system. Carry select adder (CSLA) is known to be the fastest adder among the conventional adder structures. This work presents a method to eliminate all the unnecessary logic operations present in conventional CSLA and suggest a new logic formulation for CSLA. In the suggested scheme, the selection of carry (CS) is performed before the calculation of final sum, which different from the conventional CSLA. The proposed CSLA was synthesized using Xilinx ISE and power was analyzed using Xilinx Xpower Analyzer.**

Keywords-high speed; low power; area-delay product; carry-Select-adder.

I. INTRODUCTION

A ripple carry adder (RCA) uses a simple design, but carries Propagation delay (CPD) is more 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 to carry inputs (c_{in} =0 and c_{in} =1) and selects one out of each pair for final-sum and final-output-carry using the control signal c_{in}. 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, Chang [4] 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 [1] 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. The CBL-based CSLA 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. However, the CBL-based SQRT-CSLA design requires more logic resource and delay than the BEC-based SQRT-CSLA. Logic optimization largely depends on availability of redundant operations in the formulation, where as adder delay mainly depends on data available at the input. In the existing design, logic is improved without giving any consideration to the data dependence. An analysis made 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, a logic formulation proposed for the CSLA. The main contribution in this is logic formulation based on data dependence and optimized carry generator (CG) and CS design.

II. CONVENTIONAL CSLA AND ITS LOGIC FORMULATION

The conventional CSLA consists of one sum and carry generation unit and sum and carry selection unit as shown in Fig. 1. Sum and carry generation unit can be composed of two ripple carry adders one with carry input zero and other with carry input one as shown in fig.2. Where n is the adder bit-width. An n-bit RCA can be composed using *half-sum* generator (HSG), *half-carry* generator (HCG), *full-sum* generator (FSG), *full-carry* generator (FCG) (shown in fig.3). The RAC-1 and RAC-2 generates n-bit sum (s^0 and s^1) and carry out ($c^o{}_{out}$ and $c^1{}_{out}$) corresponds to input-carry (c_{in}=0 and c_{in}=1) respectively.

Logic expression of RAC-1 are given as,

$$S^0{}_0(i) = A(i) \oplus B(i) \quad (1a)$$

$$C^0{}_0(i) = A(i).B(i) \quad (1b)$$

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

$$C^0{}_1(i) = c^0{}_0(i) + s^0{}_0(i).c^0{}_1(i-1) \quad (1d)$$

$$C^0{}_{out} = c^0{}_1(n-1) \quad (1e)$$

Logic expression of RCA-2 are given as,

$$S^1{}_0(i) = A(i) \oplus B(i) \quad (2a)$$

978-1-4799-6085-9/15/$31.00 ©2015 IEEE

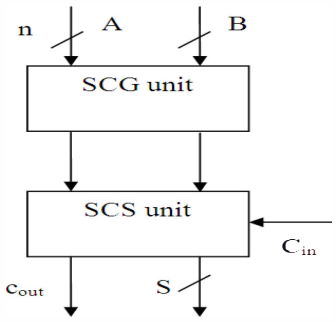

Fig.1.structure of conventional CSLA

$$C^1_0(i) = A(i) \,.\, B(i) \quad (2b)$$

$$S^1_1(i) = s^1_0(i) \oplus c^1_1(i-1) \quad (2c)$$

$$C^1_{out} = c^0_1(n-1) \quad (2e)$$

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

Here, $s^0_0(i)$, $c^0_0(i)$, $s^0_1(i)$, $c^0_1(i)$ are out of HSG, HCG, FSG, FCG respectively and c^o_{out} final carry-out of n-bit RAC-1, $s^1_0(i)$, $c^1_0(i)$, $s^1_1(i)$, $c^1_1(i)$ are out of HSG, HCG, FSG, FCG respectively and c^1_{out} final carry-out of n-bit RAC-2. From the above logic expression it is clear that (1a)-(2a) and (1b)-(2b) are identical. The HSG and HCG units can be removed from RCA-2 to have an optimized design for CSLA; here the HSG and HCG of RAC-1 can be shared to construct RCA-2. Based on this,[5] and [4] have used an add-one circuit instead of RCA-2 for the design of CSLA. A BEC based circuit is used in [1] to construct CSLA.

III. LOGIC EXPRESSION OF BEC-BASED CSLA

BEC-based CSLA consists of binary to execess-1 converter in the place of RCA-2 in conventional CSLA as shown in Fig.4. The RCA is same as that of RAC-1 in the conventional CSLA; it calculates n-bit sum $\{s^0_1(i)\}$ and carry-out $\{c^o_{out}\}$ corresponds to $c_{in} = 0$. Inputs to the BEC unit is $\{s^0_1(i), c^o_{out}\}$ and output is (n+1)-bit excess-1 code. The most significant bit (MSB) of the output of BEC unit represents c^1_{out} and remaining n least significant bits (LSBs) represent s^1_1.

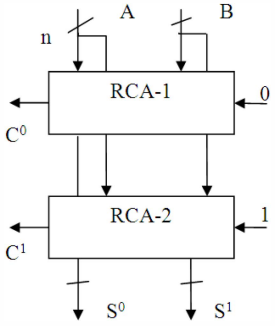

Fig.2.Structure of SCG unit

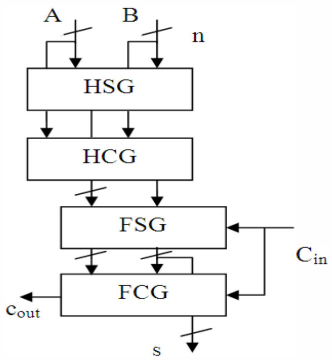

Fig.3. Structure of |Ripple Carry Adder

The logic expressions of BEC-based CSLA is given as,

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

$$C^1_1(0) = s^0_1(0) \quad (3b)$$

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

$$C^1_1(i)\, c = s^0_1(i).c^1_1(i-1) \quad (3d)$$

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

(For $1 \le i \le n-1$).

From (3d) we can say that c^1_1 depends on s^0_1, in the case of conventional CSLA c^1_1 has no dependence on s^0_1. Therefore BEC based CSLA has much data dependant compare to conventional CSLA. From (1c)-(2c) it can be observed that 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). In addition to that c^0_1 and c^1_1 depend on $\{s_0, c_0, c_{in}\}$. since c^0_1 and c^1_1 have no data dependence on s^0_1 and s^1_1, the logic expression of c^0_1 and c^1_1 can be calculated before calculating s^0_1 and s^1_1, and we can select any one from set(s^0_1, s^1_1) using selection unit for the final sum of CSLA. Most of the logic resources of CSLA are spent on calculating $\{s^0_1, s^1_1\}$.Rejecting one sum-word is not efficient approach after calculation of two *sum*s. Instead of this we can select any one of carry words $\{c^0_1, c^1_1\}$ using carry input c_{in} for calculating final sum.

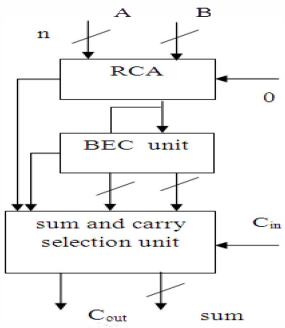

Fig.4. Structure of BEC-based CSLA

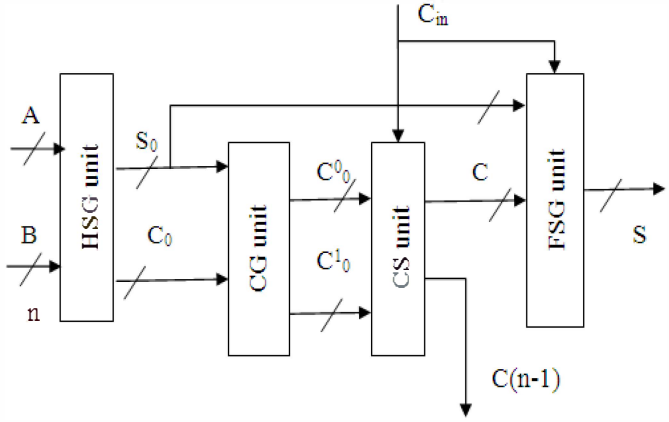

Fig.5. proposed CSLA structure

The selected carry word {i.e. either c^0_1 or c^1_1} is added with the *half-sum* to calculate the *final-sum*. On doing so we can avoid calculation of s^0_1; we can use n-bit select unit instead of using (n+1)-bit select unit; and, output-carry delay is small. Therefore we can remove all redundant logic operations present in (1a)-(1e) and (2a)-(2e) and we can rearrange based on their data dependence.

IV. PROPOSED CSLA DESIGN AND FORMULATION

The proposed CSLA structure is as shown in Fig.5. It is composed of one half-sum generation (HSG) unit, one full-sum generation (FSG) unit, one carry-generation (CG) unit, and one carry-selection (CS) unit. The CG unit composed of two units namely CG0 and CG1 Corresponding to input-carry '0' and '1', respectively[6]. Input to the HSG unit is two n-bit operands A and B and outputs are *half-sum* (HS) word S_0 and *half-carry* (HC) word C_0 of width n-bit each. CG unit receives both S_0 and C_0 from HSG unit and gives two n-bit full-carry words c^0_1 and c^1_1 corresponds to carry-input '0' and '1',respectively. The carry selection unit selects final carry based on the c_{in} from two anticipated carry words c^0_1 and c^1_1. If $c_{in} = 0$ then it selects c^0_1; otherwise it selects c^1_1. C_{out} is the MSB of c obtained from CS unit and remaining (n-1) LSBs of CS unit are XORed with (n-1) MSBs of half-sum (s_0) in the FSG unit to obtain final-sum. The proposed logic formulation for the CSLA is given by

$$S_0(i) = A(i) \oplus B(i) \quad (4a)$$

$$C_0(i) = A(i).B(i) \quad (4b)$$

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

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

$$C(i) = c^0_1(i) \quad \text{if (cin = 0)} \quad (4e)$$

$$C(i) = c^1_1(i) \quad \text{if (cin = 1)} \quad (4f)$$

$$C_{out} = c(n\text{-}1) \quad (4g)$$

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

V. PERFORMANCE COMPARISON B/W DIFFERENT TYPES OF CSLAS

A. *Estimation of Area-Delay*

Area and delay of the overall design is calculated by using the following relations:

$$\text{Area} = n.N_n + o.N_o + a.N_a \quad (5a)$$

$$\text{Delay} = n_n.T_n + n_o.T_o + n_a.T_a \quad (5b)$$

Where (T_n, T_o, T_a) and (n, o, a), represents the delay and area of (NOT, OR, AND) gates respectively. (N_n, N_o, N_a) and

TABLE-I AREA of AND, OR, NOT gates in the SAED 90-nm standard cell library datasheet

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

(n_n, n_o, n_a) represents number of gate count and critical path of (NOT, OR, AND) gates of the total design.In this design each gate is made by using 2-input AND, 2-input OR, and NOT. Two inputs XOR is composed of two AND, one OR, and two NOT gates. Area and delay of AND, OR, NOT gates which is taken from the synopsis Armenia Educational Department 90-nm standard cell library datasheet for theoretical estimation of area and delay of the design(shown in table-I). Each gate in the design is realized by using AND-OR-NOT gates. 2-input XOR gate is realized by using two 2-input AND gates, two NOT gates and one OR gate. Area-Power product (APP) and Area-Delay-Power product of conventional CSLA [13], BEC-based CSLA [1], CBL-based CSLA [2], proposed CSLA is compared in the fig.6. From that graph it is clear that ADP of CBL-based CSLA is much more than other type of CSLA, as number of bits increased ADP of CBL-based CSLA differ more from its counterpart. ADPP of Conventional CSLA is increased more than that of Prop. CSLA.

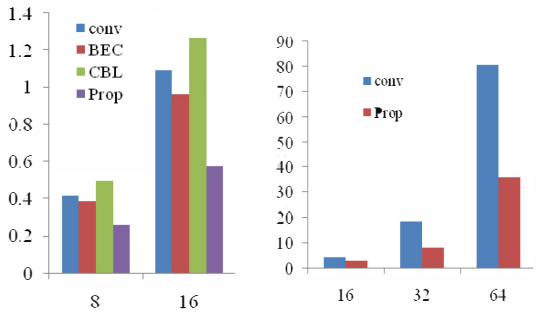

Fig.6 (a). Comparison of APP Fig.6 (b). Comparison of ADPP

TABLE-II Comparison of Area and Delay complexities between proposed and existing CSLAs

Design	width	Area (μm^2)	Power (mw)	Delay (ns)	ADP	EADP (%)
CONV	8	1438.12	288	2.27	3.56	36.97
	16	2934.28	371	4.211	12.36	55.28
CSLA (BEC)	8	1282.45	299	1.979	2.54	6.72
	16	2587.01	372	3.441	8.90	11.80
CSLA (CBL)	8	1654.65	299	2.635	4.36	83.19
	16	3416.17	370	5.225	17.85	124.25
CSLA (prop)	8	951.09	270	2.504	2.38	----
	16	1924.30	297	4.138	7.96	----

CONV: conventional, ADP: area delay product, EADP: excess ADP over the proposed design.

From the table-II we can say that proposed CSLA design is much better than the existing CSLA designs. RCA-RCA based CSLA is having 55.28% excess ADP; BEC based CSLA having 11.80% and CBL based CSLA having 124.25% excess ADP over proposed CSLA for 16-bit width.

VI. CONCLUSION

In this work we removed all the unnecessary logic operations present in conventional CSLA and proposed a new logic formulation for CSLA. In the suggested model, the selection of carry (CS) is performed before the calculation of final sum, which different from the conventional CSLA. Carry selection unit can be implemented by using AND-OR gates instead of multiplexer since it follows specific bit pattern. Proposed CSLA design is much better than the existing CSLA designs. RCA-RCA based CSLA is having 55.28% excess ADP; BEC based CSLA having 11.80% and CBL based CSLA having 124.25% excess ADP over proposed CSLA for 16-bit width. The proposed CSLA was synthesized using Xilinx ISE design suit 14.7 and power was analyzed using Xilinx Xpower Analyzer.

REFERENCES

[1] B.Ramkumar,H.M.Kittur,"Lower–power and Area-Efficient Carry-Select Adder", IEEE trans. Very large scale integr. (VLSI) syst., vol. 20, no.2, pp.371-375, Feb.2012.

[2] S.Manju and V.Sorangopal, "An efficient SQRT architecture of Carry Select Adder design by Common Boolean logic",in proc.VLSI ICEVENT, 2013.

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

[4] 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.

[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] Basant Kumar Mohanty, and Sujit Kumar Patel "Area-Delay- Power Efficient Carry-Select Adder",IEEE transaction on circuits and systems, VOL.61, NO.6,JUNE 2014.

[7] C.Nagendra, M.J.Irwin, and R.M. Owens, "Area -Time-Power tradeoffs in parallel adders", Trans. Circuits Syst. II, vol.43, pp.689– 702 Oct.1996.

[8] Sauvagya Ranjan Sahoo, Kamala Kanta Mahapatra, "Design of Low Power and High Speed Ripple Carry Adder Using Modified Feedthrough Logic" in 2012 International Conference on Communications, Devices and Intelligent Systems (CODIS) , 978-1-4673-4700.

[9] R.W.Doran, "Variants of an improved carry-lookahead adder," IEEE Trans. Computers, vol. 37,Sep.1988.

[10] T.Kim, W.Jao, and S. Tjiang, "Arithmetic optimization using carry-save-adders", in Proc.Design Automation Conf., Jun. 1998.

[11] Mark Krill, "Architecture Comparison of Multiple Operand Adders: 16 Operand8-Bit Wallace Tree v.s. Dadda Tree", Pennsylvania State University, 28 February 2003.

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

[13] O.J.Bedrij, "Carry-Select Adder," IRE Trans. Electron. Comput.,vol. EC-11, no. 3, pp. 340–344, Jun. 1962.