ASIC IMPLEMENTATION OF 16 BIT CARRY SELECT ADDER

Nomula Poojaramani[1], A.Vikram[2]

[1] Student, Sree Chaitanya Institute Of Tech. Sciences, Karimnagar, Telangana, INDIA
[2] Assistant Professor, Sree Chaitanya Institute Of Tech Sciences, Karimnagar, Telangana, INDIA

ABSTRACT

Adders are the basic building blocks of any processor or data path application. In adder design carry generation is the critical path. To reduce the power consumption of data path we need to reduce Area of the adder. Carry Select Adder is one of the fast adder used in may data path applications. The proposed design is implemented without using multiplexer and RCA structure with Cin=1. Instead of multiplexer and RCA Cin=1 structure hear we used simple combinational circuit which consists AND and XOR gates. In this proposed design has reduced Area and power as compared with regular CSLA and modified BEC(Binary to Excess-One Converter) CSLA with slight increase in the delay. . 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 16-bit 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.

Keyword: - *CSLA, BEC, LOW POWER, ADDER.*

1. INTRODUCTION

Design of area and power-efficient high speed data path logic systems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary 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 many computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum [1]. However, the CSLA is not area efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input Cin = 0 and Cin = 1, then the final sum and carry are selected by the multiplexers (mux).

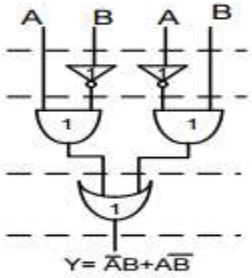


Fig-1: delay and Area evaluation of an XOR gate

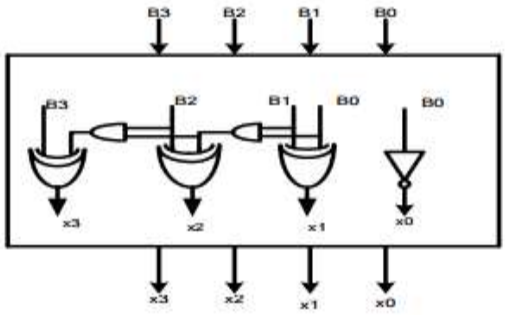


Fig-2: 4-bit BEC (Binary to Excess-One Converter)

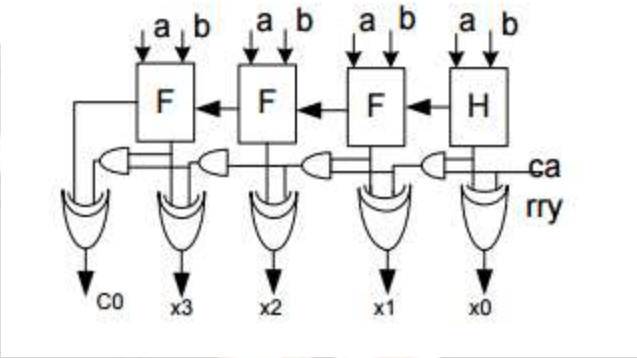


Fig-3: 4-bit CSLA without using Multiplexer

2. LITERATURE REVIEW

Bedriji 1962 proposes that the problem of carry propagation delay is overcome by independently generating multiple radix carries and using these carries to select between simultaneously generated sums. AkhilashTyagi 1993 introduces a scheme to generate carry bits with block carrying 1 from the carries of a block with block carrying 0 Chang and Hsiao 1998 propose that instead of using dual carry ripple adder a carry select adder scheme using an add one circuit to replace one carry ripple adder Youngwood Kim and Lee Sup Kim 2001 introduces a multiplexer based add one circuit is proposed to reduce the area with negligible speed penalty. Yajuan He et al 2005 proposed an area efficient square root carry select adder scheme based on a new first zero detection logic. Ramkumar and Harish 2012 propose BEC technique which is a simple and efficient gate level modification to significantly reduce the area and power of square root CSLA. Padma Devi et al 2010 proposed modified carry select adder designed in different stages which reduces the area and power consumption

3. 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.

3.1 Logic Expressions of the SCG Unit of the Conventional CSLA

As shown in Fig. 4(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 (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

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

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

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

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

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

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

where $c_0^{\,1}(-1) = 0$, $c_1^{\,1}(-1) = 1$, and $0 \le i \le n-1$.

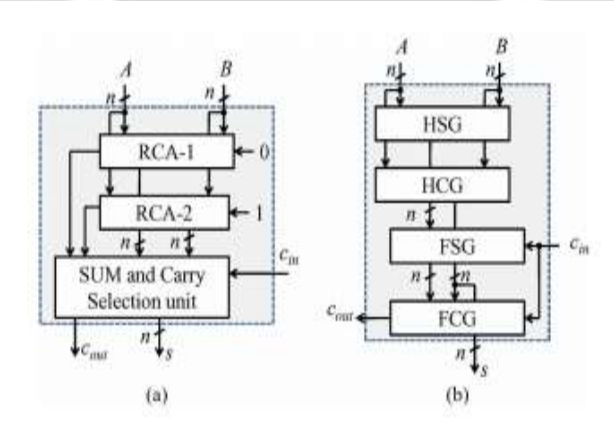


Fig-4: (a) Conventional CSLA; n is the input operand bit-width.

(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.

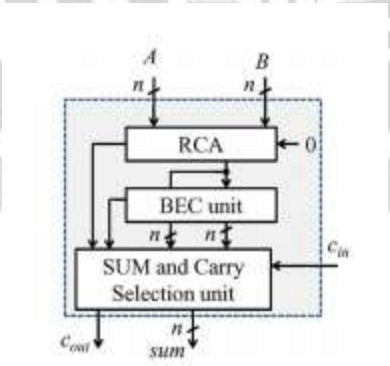


Fig-5: Structure of the BEC-based CSLA; n is the input operand bit-width.

As shown in (1a)–(1c) and (2a)–(2c), the logic expression of {s0 0(i), c0 0(i)} is identical to that of {s1 0(i), c1 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.

3.2 Logic Expression of the SCG Unit of the BEC-Based CSLA

As shown in Fig. 5, the RCA calculates n-bit sum s0 1 and c0 out corresponding to cin = 0. The BEC unit receives s0 1 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 s1 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

$$s_1^1(0) = \overline{s_1^0}(0) \quad c_1^1(0) = s_1^0(0) \tag{3a}$$

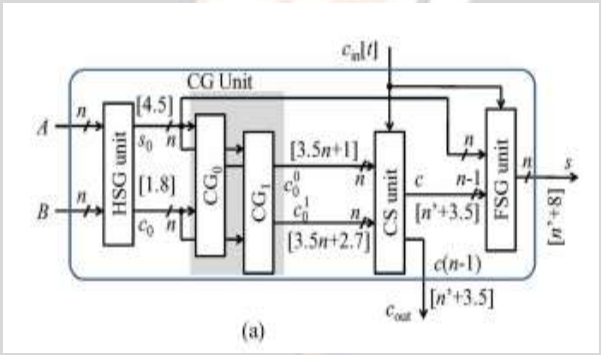
$$s_1^1(i) = s_1^0(i) \oplus c_1^1(i-1) \tag{3b}$$

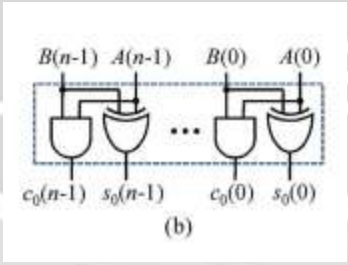
$$c_1^1(i) = s_1^0(i) \cdot c_1^1(i-1) \tag{3c}$$

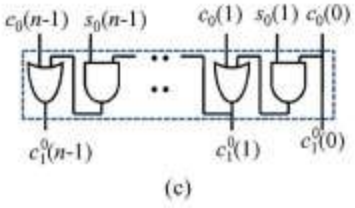
$$c_{\text{out}}^1 = c_1^0(n-1) \oplus c_1^1(n-1) \tag{3d}$$

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

We can find from (1a)–(1c) and (3a)–(3d) that, in the case of the BEC-based CSLA, c1 1 depends on s0 1, which otherwise has no dependence on s0 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.







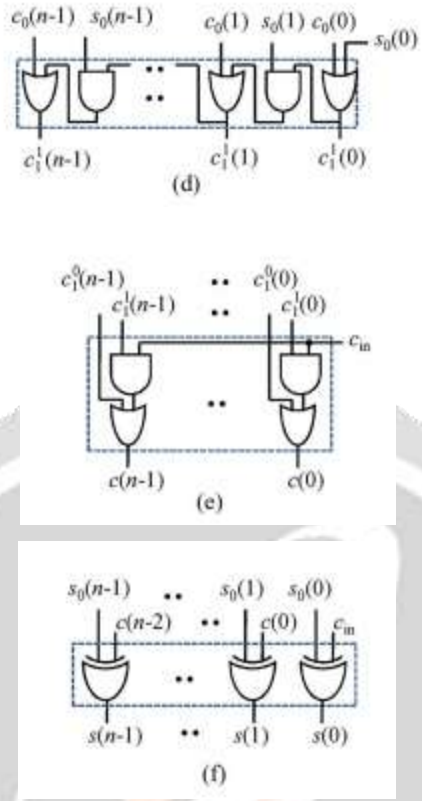


Fig-:6. (a) Proposed CS adder design, where n is the input operand bit-width, and [∗] represents delay (in the unit of inverter delay), n = max(t, 3.5n + 2.7). (b) Gate-level design of the HSG. (c) Gate-level optimized design of (CG0) for input-carry = 0. (d) Gate-level optimized design of (CG1) for input-carry = 1. (e) Gate-level design of the CS unit. (f) Gate-level design of the final-sum generation (FSG) unit.

It is interesting to note from (1a)–(1c) and (2a)–(2c) that logic expressions of s0 1 and s1 1 are identical except the terms c0 1 and c1 1 since (s0 0 = s1 0 = s0). In addition, we find that c0 1 and c1 1 depend on {s0, c0, cin}, where c0 = c0 0 = c1 0. Since c0 1 and c1 1 have no dependence on s0 1 and s1 1, the logic operation of c0 1 and c1 1 can be scheduled before s0 1 and s1 1, and the select unit can select one from the set (s0 1, s1 1) for the final-sum of the CSLA. We find that a significant amount of logic resource is spent for calculating {s0 1, s1 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 {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 s0 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

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

4. PROPOSED ADDER DESIGN

The proposed CSLA is based on the logic formulation given in (4a)–(4g), and its structure is shown in Fig. 6(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 c0 1 and c1 1 corresponding to input-carry '0' and '1', respectively. The logic diagram of the HSG unit is shown in Fig. 6(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. 6(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 cin. It selects c0 1 when cin = 0; otherwise, it selects c1 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 c0 1 and c1 1 follow a specific bit pattern. If c0 1(i) = '1', then c1 1(i)=1, irrespective of s0(i) and c0(i), for $0 \leq i \leq n-1$. 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 cout, and (n − 1) LSBs are XOR ed with (n − 1) MSBs of half-sum (s0) in the FSG [shown in Fig. 6(f)] to obtain (n − 1) MSBs of final-sum (s). The LSB of s0 is XOR ed with cin to obtain the LSB of s.

5. SIMULATION RESULTS

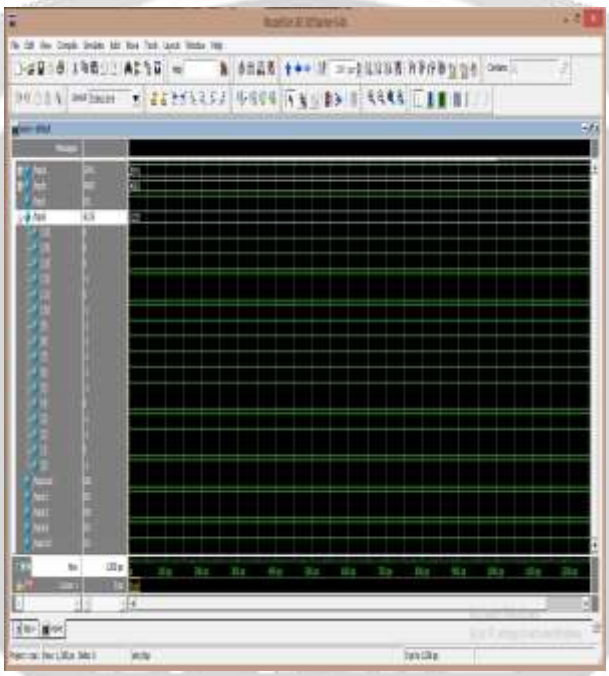


Fig-7: Simulation result for 16-bit proposed carry select adder

4. CONCLUSIONS

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 16-bit CSLA design involves significantly less area and delay than the recently proposed BEC-based CSLA. Due to the small carry output delay, the proposed 16-bit CSLA design is a good candidate for the SQRT adder.

6. 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 carryselect 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.