

An Overview of Delay-Area Efficient Carry-Select Adder

Ranjit Chandrakant Gawande¹, Prof Vinodkumar P. Patil ²

PG Scholar, Dept. of Electrical Engg, NES's Gangamai College of Engineering, Nagaon, M.S., India¹

Assistant Professor, Dept. of Electrical Engg, NES's Gangamai College of Engineering, Nagaon, M.S., India²

Abstract- An adder with superb area & speed is one of the most main areas of study in VLSI system design. With finest area & speed, reducing the power consumption is also main area of research in VLSI system design. Here we use a carry select adder design for the realization of fast adder. There are several choices for design of carry select adder. Carry select adder (CSLA) is one of the fastest adders and used in many data processing processors to achieve fast arithmetic function. From the construction of the CSLA, we come to know that there is an choice for reducing the area and power consumption in the CSLA. This work uses an easy and planned gate-level adjustment to considerably reduce the area and power consumption of CSLA. We compare some of these techniques with existing conventional fast adder design to verify its efficiency.

Keywords- VLSI system design, CSLA, power consumption, conventional fast adder.

I. INTRODUCTION

Design of high speed digital adders with well-organized area and power is one of the most important areas of study in VLSI system design. Adders are the central component of microprocessors and digital signal processors. The Ripple Carry Adders (RCAs) have the best design by the area point of view between all types of adders; but they are the slowest types of adders. On the other hand, Carry Look-ahead Adders (CLAs) are the fastest adders, but there area requirement is large. Carry Select Adders have the advantages of both RCAs and CLAs, as it has the best area like RCAs and the short delay like CLAs. In digital adders, the speed of addition is limited by the time required to circulate a carry through the adder. The calculation designed for every bit position in a basic adder is formed in order only after the previous bit position has been summed and a carry circulated into the next position.

The CSLA used in a lot of mathematics systems to solve the difficulty of carry circulation delay by independently generating several a carry and then choose a carry to create the final sum [1] [2]. On the other hand, the CSA is not area efficient as it uses a number of pair of adders to generate a half-done sum and carry by allowing for carry input $C_{in}=0$ and $C_{in}=1$, after that the final sum and carry are selected by the mux [1]. The carry out bit of the previous block of the adders acts as the pick signal to the mux. A number of examples of such adders have been published and there are several efficient executions.

II. LITERATURE SURVEY

As we see adders have fundamental importance in a wide variety of digital systems, a lot of fast adders are present today, but adding fast using low area and power is still difficult. In digital adders, the speed of addition is limited by the time required to spread a carry through adder. So the CSLA is used in many computational systems to recover the problem of carry propagation delay. Thus many papers were published on this through a number of examples of such adders and many capable implementations were also done.

In 1962, O.J.Bedrij [1] described the extremely fast digital adder with sum selection and multiple-radix carry. He reviewed the number of hardware and the logical delay for a 100-bit ripple-carry adder and a carry-select adder. The complexity of carry-propagation delay was solved by separately generating multiple-radix carries and using these carries to select among all together generated sums. In this adder structure, the addend and augend were separated into subaddend and subaugend sections that were added two times to create two sub sums. One addition was done with a carry digit forced into each section, and the other addition combined the operands without the forced carry digit. The choice of the accurate sub sum from each of the adder sections depended upon whether or not there really was a carry into that adder section.

There are many carry select adder approaches available but most of them use ripple carry adder. T.Y.Chang and M.J.Hsiao [3], suggested that instead of using dual carry-ripple adders, use a carry select adder method using a add one circuit to replace one carry-ripple adder which requires 29.2% fewer transistors with a speed penalty of 5.9% for bit length $n=64$. If speed was important for this 64 bit adder, then two of the original carry-select adder blocks could be replaced with the proposed scheme with a 6.3% area saving and the same speed.

The Youngjoon kim and Lee-Sup Kim [4] suggested that a carry-select adder could be implemented by using single ripple carry adder and an add-one circuit instead of using dual ripple-carry adders. They proposed a new add-one circuit using the first zero finding circuit and multiplexers to reduce the area and power with no speed penalty. For bit length $n=64$, this new carry-select adder requires 38 percent fewer transistors than the dual ripple-carry carry-select adder and 29 percent fewer transistors than Chang's carry-select adder using single ripple carry adder. This new 64b adder had 3.45 ns delay time at 2.5 V power supply using a 0.25 μm CMOS technology. Behnam Amelifard et.al [6], suggested a new adder called carry select adder with sharing (CSAS) which was area efficient but the delay was more. M.Alioto et.al [5], suggested using variable size block sizing depending on the multiplexers delay.

The B.Ramkumar, H.M.Kittur, and P. M. Kannan [7] suggested a very simple approach to improve the speed of addition. Based on this approach a 16, 32 and 64-bit adder architecture was developed and compared with conventional fast adder architectures. In a number of parallel multipliers to speed up the final calculation, CLA was set in the form of Carry Select adder (CSLA) & was used. But due to the structure of the CSLA it occupied more chip area, because it uses multiple pairs of RCA's to generate the partial sum and carry by considering $C_{in}=0$ and $C_{in}=1$. Thus the complexity of the final adder structure was high. So they replaced the RCA (CLA) with $C_{in}=1$ with BEC logic, which reduced the maximum area and delay in the final adder structure.

III. LOGIC FORMATION

The CSLA has two parts:

- 1) The sum and carry generator unit (SCG)
- 2) The sum and carry selection unit

The SCG unit utilizes the most part of the logic resources of CSLA and greatly contributes to the critical path. Different logic designs have been recommended for well-organized implementation of the SCG unit. We prepared a study of the logic designs suggested for the SCG unit of conventional and BEC-based CSLAs of by proper logic terms. The main aim of this study is to discover unwanted logic operations and data dependence.

Logic terms of the SCG Unit of the Conventional CSLA:

As shown in Fig.1 (a), the SCG unit of the conventional CSLA [1] is made of two n -bit RCAs, where n is the adder bit-width. The logic operation of the n -bit RCA is performing 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 creates n -bit sum (s_0 and s_1) and output-carry (c_{out}^0 and c_{out}^1) corresponding to input-carry ($c_{in} = 0$ and $c_{in} = 1$), respectively.

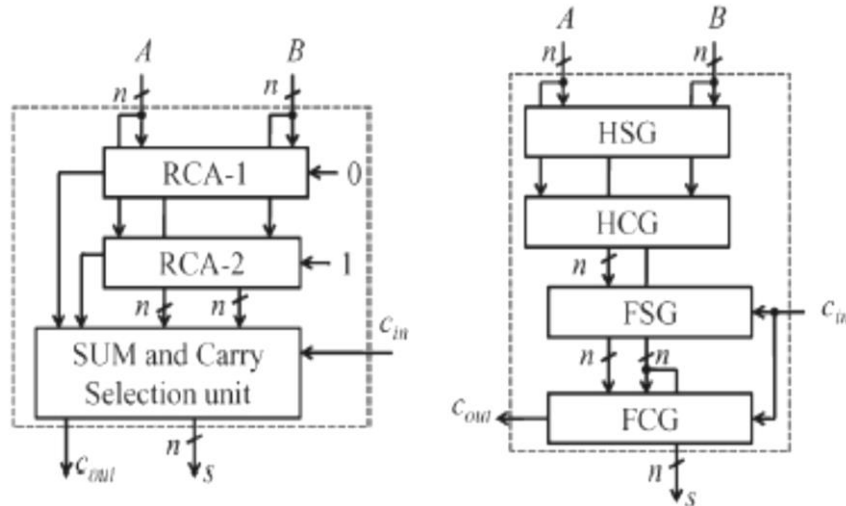


Fig.1 (a) Conventional CSLA; n is the input operand bit-width. (b) The logic operations of the RCA

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.

Logic terms of RCA-1 and RCA-2 of the SCG unit of then-bit CSLA are given as

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

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

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

$$s_1^1(i) = A(i) \oplus B(i) \quad c_1^1(i) = A(i) \cdot B(i) \quad \text{-----} \quad (2a)$$

$$s_1^1(i) = s_1^0(i) \oplus c_1^1(i-1) \quad \text{-----} \quad (2b)$$

$$c_1^1(i) = c_1^0(i) + s_1^0(i) \cdot c_1^1(i-1) \quad c_{out}^1 = c_1^1(n-1) \quad \text{-----} \quad (2c)$$

Where $c_1^0(-1) = 0$, $c_1^1(-1) = 1$, and $0 \leq i \leq n-1$.

As shown in (1a) – (1c) and (2a) – (2c), the logic term of $\{s_0^0(i), c_0^0(i)\}$ is the similar to that of $\{s_1^0(i), c_1^0(i)\}$. These unnecessary logic functions can be removed to have a better design for RCA-2, in which the HSG and HCG of RCA-1 is shared to construct RCA-2. Based on this, [4] have used an add-one circuit instead of RCA-2 in the CSLA, in which a BEC circuit is used in [8] for the same purpose. Whereas the BEC-based CSLA suggests the most excellent area, delay and power efficiency among the existing CSLAs, we talk about here the logic conditions of the SCG unit of the BEC-based CSLA as well.

Logic terms of the SCG Unit of the BEC-Based CSLA:

As shown in Fig.2, the RCA calculates n -bit sum s_1^0 and c_{out}^0 related to $c_{in} = 0$. The BEC element gets s_1^0 and c_{out}^0 from the RCA and creates $(n + 1)$ -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 terms of the RCA are the same as those given in (1a)–(1c).

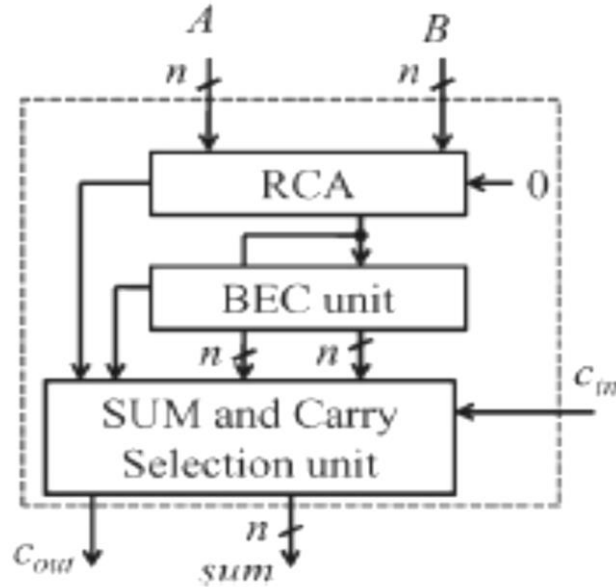


Fig.2 Structure of the BEC-based CSLA; n is the input operand bit-width.

The logic terms of the BEC unit of the n -bit BEC-based CSLA are given as

$$s_1^1(0) = s_1^0(0) \quad c_1^1(0) = s_1^0(0) \quad \text{-----} \quad (3a)$$

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

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

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

IV. SUGGEST METHOD

We can get from (1a)–(1c) and (3a)–(3d) that, in the case of the BEC-based CSLA, c_1^1 depends on s_1^0 , whereas in conventional CSLA has no dependence on s_1^0 . The BEC method consequently increases data dependence in the CSLA. We have considered logic terms of the conventional CSLA and made a more study on the data dependence to find a most excellent logic term for the CSLA. It is interesting to note from (1a)–(1c) and (2a)–(2c) that logic terms of s_1^0 and s_1^1 are the same but the terms c_1^0 and c_1^1 since ($s_1^0 = s_1^1 = s_0$). Also, we find that c_1^0 and c_1^1 depend on $\{s_0, c_0, c_{in}\}$, where $c_0 = c_1^0 = c_1^1$. Since c_1^0 and c_1^1 have no dependence on s_1^0 and s_1^1 , the logic operation of c_1^0 and c_1^1 can be planned before s_1^0 and s_1^1 , and the select unit can select one from the set (s_1^0, s_1^1) for the final-sum of the CSLA. We discover that a maximum number of logic terms is used up for calculating $\{s_1^0, s_1^1\}$, and it is not a well-organized approach to reject one sum-word after the calculation. As an alternative, one can select the required carry term from the expected carry terms $\{c_0$ and $c_1\}$ to calculate the final-sum. The chosen carry expression is added with the half-sum (s_0) to generate the final-sum (s).

Using this method, one can have three design advantages:

- Calculation of s_1^0 can be avoided in the SCG unit.
- The n-bit select unit can be used as an alternative for the $(n + 1)$ bit.
- Small output-carry delay.

V. CONCLUSION

In this paper, a variety of Carry select adder designs have been studied from the most current and previous published research work. A variety of different logics have been used till now, shown in this paper, to build the carry select adder to reduce the power, delay, area and power-delay product and transistors count. Based on this analysis it is conclude that the logic functions engaged in conventional carry select adder (CSLA) and BEC based carry select adder (CSLA) i.e., binary to excess-1 converter based carry select adder have the data dependence and unnecessary logic functions. We can remove all the unnecessary logic functions included in the conventional CSLA and offer a new logic formation for CSLA. In the propose idea, the carry select (CS) function can planned earlier than the calculation of final-sum which will be different from the conventional approach. A capable CSLA design can be obtain by using better logic units. The propose carry select adder (CSLA) design will help to reduce more area and delay than the recently proposed BEC-based carry select adder.

REFERENCES

- [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] 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.
- [8] B. Ramkumar and H.M. Kittur, "Low Power and Area Efficient Carry select Adder" *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* Volume: 20, Issue: 2, 2012.