Comments

Comments on "Self-Checking Carry-Select Adder Design Based on Two-Rail Encoding"

Muhammad Ali Akbar and Jeong-A Lee, *Senior Member, IEEE*

***Abstract*—We first show that self-checking presented in the above paper does not work for carry-select adder with input bits higher than 2. Then, we present a correct design and show that the resulting overhead almost doubles.**

INTRODUCTION*:* Self-checking carry-select adder (CSeA) with reduced area overhead was presented by Vasudevan *et al.* [1]. The proposed design seemed to be promising for self-checking CSeA, and therefore, the work was referred to for comparison by Alioto *et al.* [2], Belgacem *et al.* [3] and Wang *et al.* [4]. However, we find that the claim for self-checking is only valid for 2-bit CSeA, and the 6-bit CSeA shown in the paper cannot provide self-checking.

In this paper, we argue the failure of their design model and present a correct self-checking CSeA design based on two-rail encoding. We show that the transistor overhead of the correct model is higher than the one claimed by Vasudevan *et al.* [1].

DESIGN PROBLEM*:* The property used by Vasudevan *et al.* [1] for a pair of full adders can be generalized as follows:

> *The relation between sum bits calculated with identical inputs is only dependent on the carry-input, and for complemented values of carry-input, we will obtain complemented sum bits (keeping other pairs of input bits identical).*

For example, if the sum bits S_i^1 and S_i^0 are generated by using intermediate carry C_i^1 and C_i^0, respectively, then according to the above-mentioned statements:

$$\textbf{If } \left(C_i^1 == \overline{C_i^0}\right);$$

$$\textbf{Then } S_i^1 \text{ is equal to } \overline{S_i^0}.$$

Where, i indicates the bit position, while 1 and 0 in the superscript represent the initial value of Cin. If we analyze the CSeA design, then the above statements are only valid for the least significant sum bit of a particular CSeA block. This is because the first full-adder in every CSeA block is the only one that will get the complemented values of the carry-input. The remaining full adders will depend on the propagated carry, which may or may not be complementary to each other. Therefore, we cannot say whether the generated sum bits, other than the first full adder, will be inverted to each other or not.

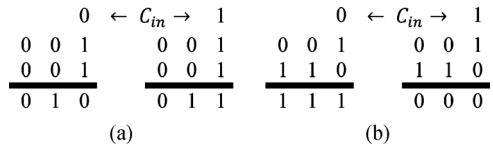

Fig. 1. Examples of 3-bit addition (a) $S_2^0 = S_2^1$ (b) $S_2^0 = \overline{S_2^1}$.

The possibility of having equal values of propagated carry by the two corresponding adders in a CSeA was neglected by Vasudevan *et al.* Therefore, the approach in [1] fails for CSeA with more than two bits, as shown in Fig. 2[1]. Note that the initial carries, C_1 and C_2, always complementary to each other because of the design requirement of CSeA, while the intermediate carries, C_a and C_b, may or may not be complementary to each other, depending on the conditions of carry propagation. Thus, S_a and S_b will not always be complementary to each other. Therefore, comparing S_a and S_b directly using 2-pair-2-rail-checker (TTRC) will give the wrong indication of faults. Even if there is no fault, the TTRC will indicate a fault. Since the problem in their approach starts from C_a and C_b, we do not discuss the intermediate carries C_x and C_y. Let us consider the binary addition of 3-bit numbers as illustrated in Fig. 1. It can easily be seen from Fig. 1(a) and (b) that the most significant bits may or may not be complementary to each other.

PROPOSED DESIGN SOLUTION*:* A carry-select adder pre-computes sum bits using two parallel ripple-carry adders (RCAs), with complemented values of the initial C_in, and the actual value of the C_in will be used to determine the final sum bit. Vasudevan *et al.* utilized both RCAs to obtain the complementary behavior of the corresponding sum bits. However, it is possible to perform a logical operation such that one of the RCA blocks should always provide inverted sum bits with respect to the opponent block for checking purposes only. This will provide a more simplified and systematic design, which can be extended easily. In this paper, we will discuss only one possible way in which the sum bits calculated at initial $\mathrm{C}_\mathrm{in} = 0$ are altered, such that they become complementary to the sum bits calculated at an initial $\mathrm{C}_\mathrm{in} = 1$ for comparison.

After close observation, we found that:

> *Except for the least significant bit, the sum bit computed when initial carry-in equals 0 will be complementary to the corresponding sum bit with an initial carry-in equal to 1 only when all the lower sum bits are equal to logic-1.*

Manuscript received July 18, 2013; revised November 10, 2013; accepted November 29, 2013. Date of publication January 24, 2014; date of current version June 24, 2014. This paper was recommended by Associate Editor M. Alioto.

The authors are with Computer System Lab, Department of Computer Engineering, Chosun University, 501-759, South Korea, (e-mail: mali.neduet@gmail.com; jalee@chosun.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2013.2295930

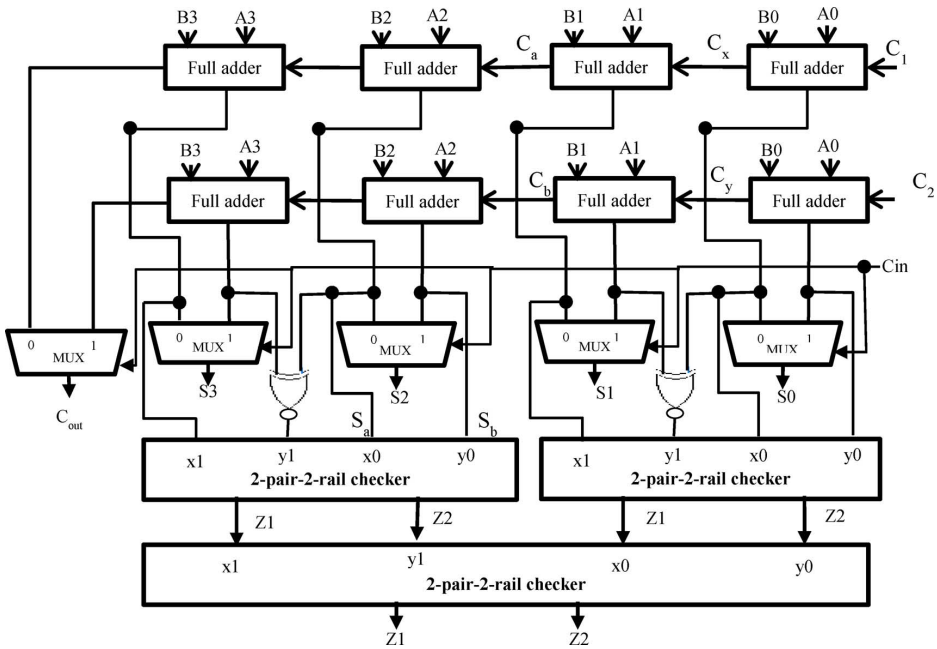

Fig. 2. Faulty design of 4-bit self-testing carry-select adder [1].

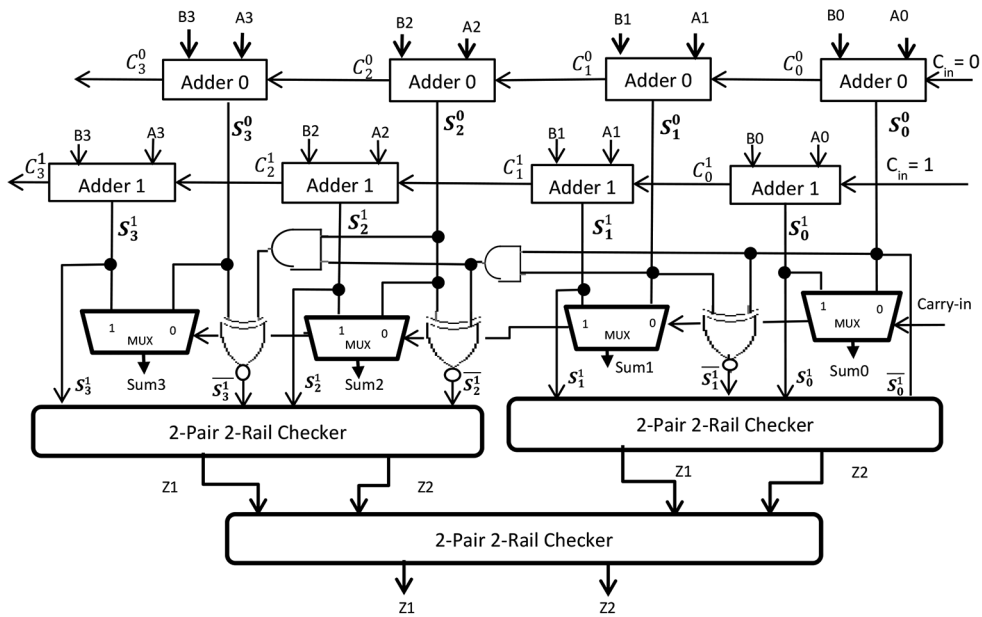

Fig. 3. Corrected design of self-testing carry-select adder with two rail encoding.

In general, we can say that:

$$\textbf{If } \left(S_1^0 \cdot S_2^0 \cdot S_3^0 \ldots\ldots S_{(i-1)}^0 = 1\right)$$
$$\textbf{Then } S_i^0 \text{ is equal to } \overline{S_i^1};$$
$$\textbf{Else } S_i^0 \text{ is equal to } S_i^1.$$

Thus, in order to apply TTRC for n-bit CSeA, we need to have S_n^1 along with its complement, and S_n^0 will not always equal to $\overline{S_n^1}$. If all the $(\mathrm{n}-1)^{\mathrm{th}}$ sum bits at $\mathrm{C_{in}} = 0$ are equal to logic-1, then the value S_n^0 is equal to the complement of S_n^1. In other cases, if any of the $(\mathrm{n}-1)^{\mathrm{th}}$ sum bits at $\mathrm{C_{in}} = 0$ are equal to logic-0, then we take the inverse of S_n^0, so it equals to the complement of S_n^1. Therefore, in (1) we performed an XNOR operation between S_n^0 and the product of all lower sum bits computed at the initial $\mathrm{C_{in}} = 0$, such that the resultant K will always be equal to $\overline{S_n^1}$. The design module shown in Fig. 3 will be used to implement the 4-bit self-checking CSeA.

$$\mathrm{K} = \overline{S_n^0 \oplus \left(S_{(n-1)}^0 \ldots\ldots S_3^0 \cdot S_2^0 \cdot S_1^0\right)}. \tag{1}$$

TABLE I
COMPARISON OF CSEA WITH SELF-CHECKING CSEA BEFORE AND AFTER CORRECTION OF VASUDEVAN *et al.* [1].

Number of bits	CSeA without self-checking	Vasudevan et al. faulty design [1]			Corrected self-checking CSeA		
	Transistor required [1]	Transistor required	Transistor overhead	% Transistor overhead	Transistor required	Transistor overhead	% Transistor overhead
4-bit	284	328	44	15.49%	350	66	23.2%
6-bit	420	490	70	16.66%	534	114	27.14%
8-bit	556	652	96	17.26%	718	162	29.14%
16-bit	1100	1300	200	18.18%	1454	354	32.18%
32-bit	2188	2596	408	18.65%	2926	738	33.73%
64-bit	4364	5188	824	18.88%	5870	1506	34.5%

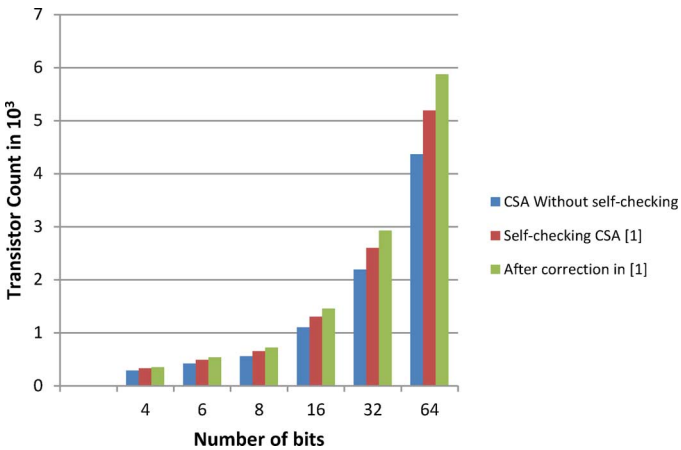

Fig. 4. Comparison with CSeA, self-checking CSeA [1] and our proposed solution.

COMPARISON*:* We applied the same technology and implementation used by Vasudevan *et al.* [1] for comparison. A standard complementary metal-oxide semiconductor-based AND gate with 6 transistors was used for area computation and the transistor count for full-adder, multiplexer (MUX), XNOR gate and TTRC was taken from Vasudevan *et al.* [1], as given below:

- Full adder – 28 transistors;
- MUX – 12 transistors;
- XNOR – 10 transistors;
- TTRC – 8 transistors.

For an n-bit self-checking CSeA, we required $(n-2)$ number of AND gates, $(n-1)$ number of XNOR gates, $(n+1)$ number of MUX, $(2n)$ number of full adders and $(n-1)$ number of TTRC, respectively. From Table I, we can see that the difference in transistor overhead for 4- to 64-bit self-checking CSeAs varies from 22 to 682, compared to the faulty self-checking CSeA design by Vasudevan *et al.* [1]. Moreover, the transistor overhead of the corrected self-checking CSeA, as compared to CSeA without self-checking, was found to be 23.2% to 34.5%, whereas in the faulty approach presented by Vasudevan *et al.* [1], the overhead was 15.49% to 18.84%.

A graphical representation for area comparison is shown in Fig. 4. We can see that after correcting the self-checking CSeA design [1], the percent change in transistor count shows an increasing trend with the increase in number of bits in the adder.

REFERENCES

[1] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, "Self-checking carry-select adder design based on two-rail encoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 12, pp. 2696–2705, Dec. 2007.

[2] M. Alioto, G. Palumbo, and M. Poli, "Optimized design of parallel carry-select adders," *Integration, the VLSI J.*, vol. 44, no. 1, pp. 62–74, Jan. 2011.

[3] H. Belgacem, K. Chiraz, and T. Rached, "A novel differential XOR-based self-checking adder," *Int. J. Electron.*, vol. 99, no. 9, pp. 1239–1261, Apr. 2012.

[4] Y. S. Wang, M. H. Hsieh, J. C.-M. Li, and C. C.-P. Chen, "An at-speed test technique for high-speed high-order adder by a 6.4-GHz 64-bit domino adder example," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 8, pp. 1644–1655, Aug. 2012.