A New Self-checking Sum-bit Duplicated Carry-select Adder*

E. S. Sogomonyan † D. Marienfeld V. Ocheretnij M. Gössel

University of Potsdam, Department of Computer Science,
Fault Tolerant Computing Group,
14439 Potsdam, Germany
E-mail: egor | dmarien | vitalij | mgoessel @cs.uni-potsdam.de

Abstract

In this paper the **first code-disjoint totally self-checking carry-select adder** *is proposed. The adder blocks are fast ripple adders with a single NAND-gate delay for carry-propagation per cell. In every adder block both the sum-bits and the corresponding inverted sum-bits are simultaneously implemented. The parity of the input operands is checked against the XOR-sum of the propagate signals. For 64 bits area and maximal delay are determined by the SYNOPSYS CAD tool of the EUROCHIP project. Compared to a 64 bit carry-select adder without error detection the delay of the most significant sum-bit does not increase. The area is* 170% *of a 64 bit carry-select adder (without error detection and not code-disjoint).*

1. Introduction

The first self-checking carry-select adder was described in [1], where a time redundant solution was proposed. Recently in [2, 3] self-checking carry-select adders are described which are not code-disjoint. In this paper we propose the first code-disjoint completely self-checking carry-select adder.

2. Proposed Sum-bit Duplicated Carry-select Adder

In Fig. 1 the proposed self-checking code-disjoint carry-select adder for 64 bits is shown. The input operands $a = a_0, \ldots, a_{63}$ and $b = b_0, \ldots, b_{63}$ are supposed to be parity encoded with the parity bits $p_a = a_0 \oplus \ldots \oplus a_{63}$ and $p_b = b_0 \oplus \ldots \oplus b_{63}$ respectively.

From the input operands the propagate signals $p_0 = a_0 \oplus b_0$, $p_1 = a_1 \oplus b_1, \ldots, p_{63} = a_{63} \oplus b_{63}$ are derived only once by the "*Propagate Generator*" which consists of 64 XOR-gates.

The XOR-sum of the propagate signals which is determined by 63 XOR-gates and which is equal to the XOR-sum $p(a \oplus b)$ of the bits of operands a and b, is compared with the XOR-sum $p_a \oplus p_b$ of the input parity bits p_a and p_b. Thus we save 64 XOR-gates.

As long as no error occurs we have $p_a \oplus p_b = p(a \oplus b)$.

The adder blocks of the 64 bit self-checking code-disjoint carry-select adder of Fig. 1 are in our design of block sizes of 8, 8, 12, 12, 12 and 12 bits. The adder blocks implement besides the corresponding sum-bits also the inverted sum-bits. The carry-out signals of the blocks are duplicated. All the propagate signals which are already checked by comparing $p(a \oplus b)$ with $p_a \oplus p_b$ are only determined once by the "*Propagate Generator*" for the duplicated blocks and we save $56 \cdot 3 + 8 = 176$ XOR-gates. The adder blocks are denoted by SDB.

The first block $SDB_1(8)$ which is not duplicated computes from the operand bits $a_{[0,7]} = a_0, \ldots, a_7$, $b_{[0,7]} = b_0, \ldots, b_7$ and from the propagate signals $p_{[0,7]} = p_0, \ldots, p_7$ the sum-bits $s_{[0,7]} = s_0, \ldots, s_7$, the inverted sum-bits $\bar{s}_{[0,7]} = \bar{s}_0, \ldots, \bar{s}_7$ and the duplicated carries $c_7 1$ and $c_7 2$ of the block.

The second block $SDB_2^0(8)$ computes for the constant carry-in signal 0 from the operand bits $a_{[8,15]} = a_8, \ldots, a_{15}$, $b_{[8,15]} = b_8, \ldots, b_{15}$ and from the propagate signals $p_{[8,15]} = p_8, \ldots, p_{15}$ the sum-bits $s^0_{[8,15]} = s^0_8, \ldots, s^0_{15}$, the inverted sum-bits $\bar{s}^0_{[8,15]} = \bar{s}^0_8, \ldots, \bar{s}^0_{15}$ and the duplicated carries $c^0_{15}1$ and $c^0_{15}2$ of the block.

The second duplicated block $SDB_2^1(8)$ computes for the constant carry-in signal 1 from the operand bits $a_{[8,15]} = a_8, \ldots, a_{15}$, $b_{[8,15]} = b_8, \ldots, b_{15}$ and from the propagate signals $p_{[8,15]} = p_8, \ldots, p_{15}$ the sum-bits $s^1_{[8,15]} = s^1_8, \ldots, s^1_{15}$, the inverted sum-bits $\bar{s}^1_{[8,15]} = \bar{s}^1_8, \ldots, \bar{s}^1_{15}$ and the duplicated carries $c^1_{15}1$ and $c^1_{15}2$ of the block.

If the carry-out signals $c_7 1 = c_7 2$ of the preceeding block $SDB_1(8)$ are equal to 0 (1) the multiplexors $MUXs_2$ and

*This paper was supported by a research grant of Intel
†Guest professor of the University of Potsdam

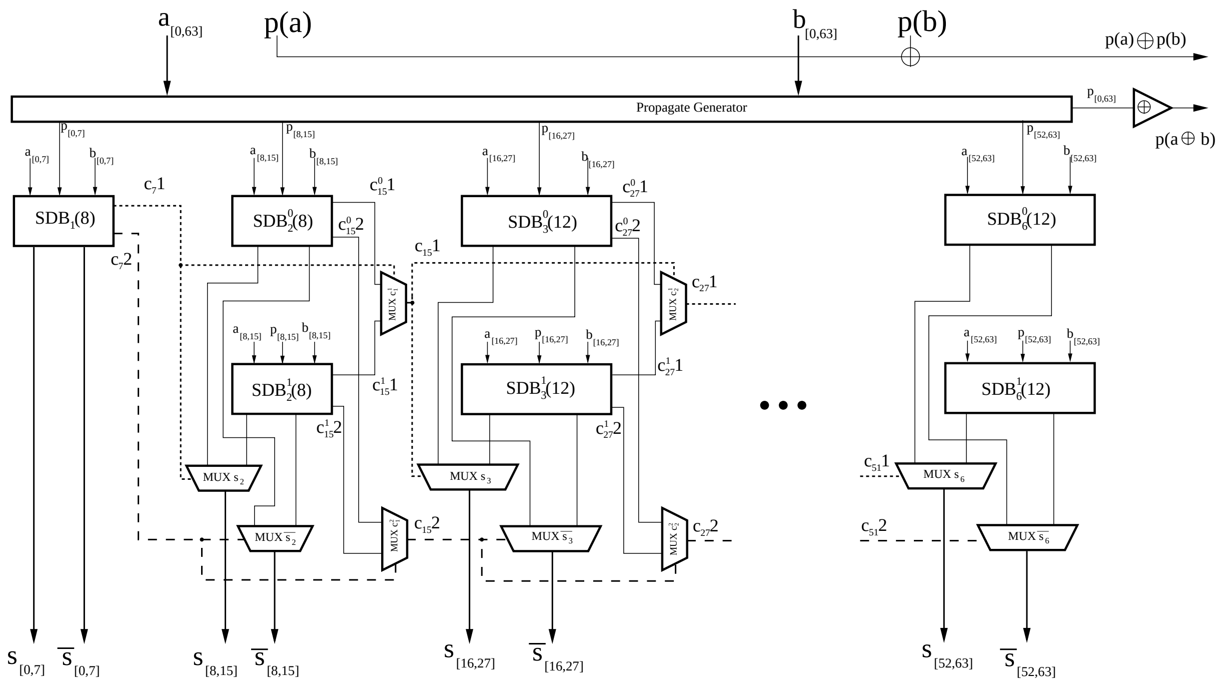

Figure 1. General structure of a 64 bit sum-bit duplicated carry-select adder

$MUX\bar{s}_2$ select $s^0_{[8,15]}$ and $\bar{s}^0_{[8,15]}$ ($s^1_{[8,15]}$ and $\bar{s}^1_{[8,15]}$) and the multiplexors $MUXc^1_2$ and $MUXc^2_2$ direct the carries $c^0_{15}1$ and $c^0_{15}2$ ($c^1_{15}1$ and $c^1_{15}2$) to their outputs. Thus we have for $c_71 = c_72 = 0$ $s_{[8,15]} = s^0_{[8,15]}$ and $\bar{s}_{[8,15]} = \bar{s}^0_{[8,15]}, c_{15}1 = c^0_{15}1$ and $c_{15}2 = c^0_{15}2$, and for $c_71 = c_72 = 1$ $s_{[8,15]} = s^1_{[8,15]}$ and $\bar{s}_{[8,15]} = \bar{s}^1_{[8,15]}, c_{15}1 = c^1_{15}1$ and $c_{15}2 = c^1_{15}2$.

In a similar way the sum-bits, the inverted sum-bits and the carry-signals of the succeeding blocks $SDB^0_3(12)$, $SDB^1_3(12)$ $SDB^0_4(12)$, $SDB^1_4(12)$; $SDB^0_5(12)$, $SDB^1_5(12)$ and $SDB^0_6(12)$, $SDB^1_6(12)$ are determined and selected by the corresponding multiplexors. All the adder blocks are implemented as fast carry-ripple adders according to [4].

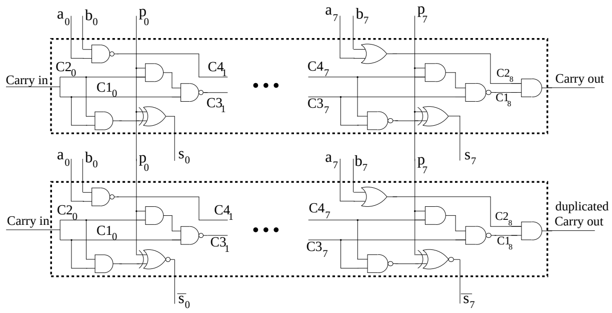

Figure 2. First sum-bit duplicated fast carry-ripple adder block

The first adder block $SDB_1(8)$ which computes $s_{[0,7]}$ and $\bar{s}_{[0,7]}$ is shown in Fig. 2. It consists of a first fast ripple adder for computing the eight sum-bits $s_{[0,7]}$ and the first carry-out signal c_71 and a second fast ripple adder with inverted outputs for computing the inverted eight sum-bits $\bar{s}_{[0,7]}$ and the duplicated carry-out signal c_72. Both these adders share the propagate signals $p_{[0,7]}$ which are derived by eight XOR-gates from the operands $a_{[0,7]}$ and $b_{[0,7]}$ and which have to be implemented only once. For details see [5]

All the adder blocks $SDB^0_2(8)$, $SDB^1_2(8), \ldots,$ $SDB^0_6(12)$, $SDB^1_6(12)$ are very similar to the sum-bit duplicated adder block in Fig. 2 with either a constant carry-in signal 0 or 1.

Compared to a completely duplicated code-disjoint carry-select adder we save 240 XOR-gates.

References

[1] F.-H. W. Shih, "High performance self-checking adder having small circuit area," in *US PS 5,018,093*, 1991.

[2] V. Ocheretnij, M. Gössel, E. S. Sogomonyan, and D. Marienfeld, "A Modulo *p* Checked Self-Checking Carry Select Adder," in *9th International On-Line Testing Symposium*, pp. 25–29, 2003.

[3] B. K. Kumar and P. K. Lala, "On-line Detection of Faults in Carry-Select Adders," in *International Test Conference (ITC)*, pp. 912–918, 2003.

[4] M. Smith, *Application-specific integrated circuits*. Adison Wesley, Reading, MA, 1997.

[5] E. S. Sogomonyan, D. Marienfeld, V. Ocheretnij, and M. Gössel, "A New Self-checking Sum-bit Duplicated Carry-select Adder." Department of Computer Science, University of Potsdam, Germany, 2003. ISSN: 0946-7580.