Self-checking Carry-select Adder with Sum-bit Duplication *

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

1 Introduction

High-speed adders are an essential part of every computer. The speed of an adder is to a large extend determined by the speed of the carry-propagation.

The main classes of adders are carry-ripple adders, carry-skip adders, carry look-ahead adders, carry-free adders and carry-select adders. Carry-select adders are the fastest of these adders.

In a carry-ripple adder the carry-out signal of every adder cell is the carry-in signal of the succeeding adder cell. Besides the carry-ripple adder all the other adders are blockwise organized. In a carry-skip adder the carry signal skips a block if all the corresponding propagate signals of this block are equal to 1. In a carry look-ahead adder the carry-in signals for the blocks are generated in a separate fast carry look-ahead unit with large area overhead. In a carry-free adder a redundant number representation is used and no carry-signal is propagated. In a carry-select adder all the blocks of the adder besides the first block, are duplicated. One of the duplicated blocks adds the corresponding bits of the operands for a constant carry-in signal 0 and the other block adds the same bits of the input operands for a constant carry-in signal 1. For a considered block the carry-out signal

*This paper was supported by a research grant of Intel

†Guest professor of the University of Potsdam

of the preceeding block selects as a control signal of a multiplexor the correct sum-bits and the correct carry-out signal as the control signal for the succeeding block. Details are described for instance in [Pa00].

All the different types of adders and mixed forms of adders are used in different fi elds of application.

Because of the shrinking dimensions in VLSI transient faults and many not modeled faults occur and are expected to increase. These faults can only be detected by concurrent checking. Self-checking circuits, and as a main part of a computer self-checking adders, are becoming more and more important. The fi rst and basic results on self-checking adder design are already described in [SHB68]. Well developed results are obtained for carry-ripple adders and carry look-ahead adders [Ni93, BNZ96, SOG01].

Till now it remains a challenging problem to design a self-checking carry-select adder.

The fi rst self-checking carry-select adder was described in [Sh91]. In [Sh91] a time redundant solution is proposed. The necessary time increases more then twice and this reduction in speed limits the fi eld of application for this adder.

Carry-select Adder with concurrent checking are considered in [OGSM03, KL03]. Both of there adders are not code-disjoint. The adder in [KL03] is not self-checking.

In this paper we propose the fi rst code-disjoint self-checking carry-select adder. The input operands are supposed to be parity encoded. The propagate signals are only once implemented for both the duplicated addder blocks with carry-in 0 and carry-in 1 and also used for checking the input parity. By this parity check also all the propagate signals are constantly monitored and all errors of the propagate signals are immediately detected. These already checked propagate signals are utilized for an optimal design of both the adder blocks with duplicated (or inverted duplicated) sum bits.

In our design the adder blocks are designed as fast carry-ripple adders as described for instance in [Sm97]. For the fast carry-ripple adder the delay for the carry propagation is only the delay of one NAND-gate per bit. Compared to a self-checking carry look-ahead adder of the same word length the proposed self-checking carry-select adder has the shortest delay for computing the 64 sum-bits of the result.

The paper is organized as follows. The proposed code-disjoint sum-bit duplicated carry-select adder is introduced in chapter 2. The adder blocks, implemented as fast carry-ripple adders are also described. In Chapter 3 it is shown that the proposed code-disjoint adder is self-testing and fault-secure with respect to single stuck-at faults. Chapter 4 contains the experimental results obtained by use of the SYNOPSYS CAD tool of the EUROCHIP project for 64 bit adders. Area and maximal delay of the proposed code-disjoint self-checking sum-bit duplicated carry-select adder are compared with the corresponding values of a carry-select adder without error detection and with a duplicated carry-select adder. In chapter 5 conclusions are drawn.

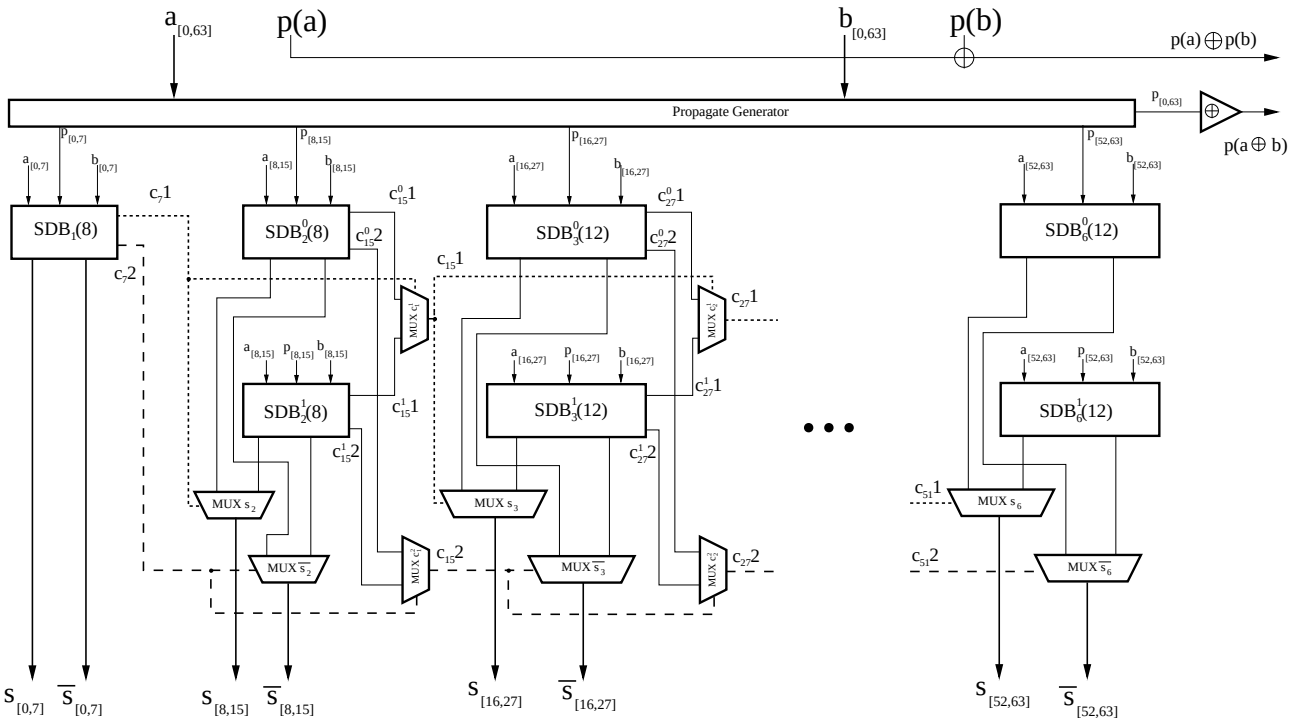

Figure 1: General structure of a 64 bit sum-bit duplicated carry-select adder

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, $p_0 \oplus p_1 \oplus \ldots \oplus p_{63} = a_0 \oplus b_0 \oplus a_1 \oplus b_1 \oplus \ldots a_{63} \oplus b_{63} = p(a \oplus b)$ 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 the corresponding sum-bits and 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 $\overline{s}_{[0,7]} = \overline{s}_0, \ldots, \overline{s}_7$ and the duplicated carries $c_7 1$ and $c_7 2$ of the block. 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. The first adder block is shown in Fig. 3.

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 $\overline{s}^0_{[8,15]} = \overline{s}^0_8, \ldots, \overline{s}^0_{15}$ 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 $\overline{s}^1_{[8,15]} = \overline{s}^1_8, \ldots, \overline{s}^1_{15}$ 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 $MUX\overline{s}_2$ select $s^0_{[8,15]}$ and $\overline{s}^0_{[8,15]}$ ($s^1_{[8,15]}$ and $\overline{s}^1_{[8,15]}$) 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 $c_7 1 = c_7 2 = 0$

$s_{[8,15]} = s^0_{[8,15]}$ and $\overline{s}_{[8,15]} = \overline{s}^0_{[8,15]}$, $c_{15}1 = c^0_{15}1$ and $c_{15}2 = c^0_{15}2$,

and for $c_7 1 = c_7 2 = 1$ $s_{[8,15]} = s^1_{[8,15]}$ and $\overline{s}_{[8,15]} = \overline{s}^1_{[8,15]}$ $c_{15}1 = c^1_{15}1$ 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 $SDB_3^0(12)$, $SDB_3^1(12)$; $SDB_4^0(12)$, $SDB_4^1(12)$; $SDB_5^0(12)$, $SDB_5^1(12)$ and $SDB_6^0(12)$, $SDB_6^1(12)$ are determined and selected by the corresponding multiplexors. All the adder blocks are implemented as fast ripple adders according to [Sm97]. Two successive adder cells of a fast ripple adder are shown in Fig. 2.

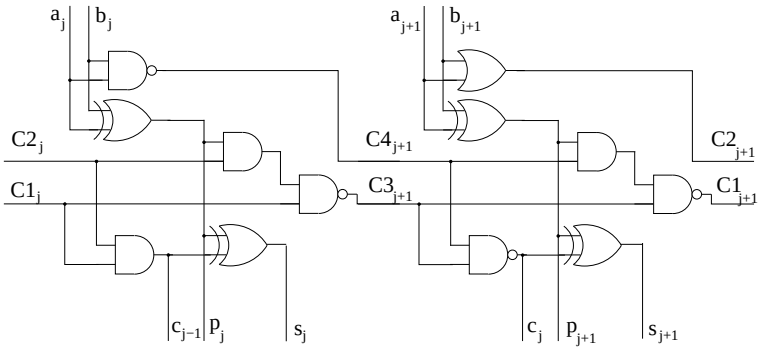

Figure 2: 2 bit fast ripple adder block

The main difference to a conventional adder is that the single carry signal c_{j-1} of the conventional ripple adder is replaced by two carry signals $C1_j$ and $C2_j$ with $c_{j-1} = C1_j \wedge C2_j$ for j even and by $C3_j$ and $C4_j$ with $c_{j-1} = \overline{C3_j \wedge C4_j}$ for j odd.

The adder is fast since the delay of the carry propagation is equal to the delay of one NAND-gate per bit only.

The first adder block $SDB_1(8)$ which computes $s_{[0,7]}$ and $\overline{s}_{[0,7]}$ is shown in detail in Fig. 3. It consists of a first fast ripple adder for computing the eight sum-bits $s_{[0,7]}$ and the first carry-out signal $c_7 1$ and a second fast ripple adder with inverted outputs for computing the inverted eight sum-bits $\overline{s}_{[0,7]}$ and the duplicated carry-out signal $c_7 2$. 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.

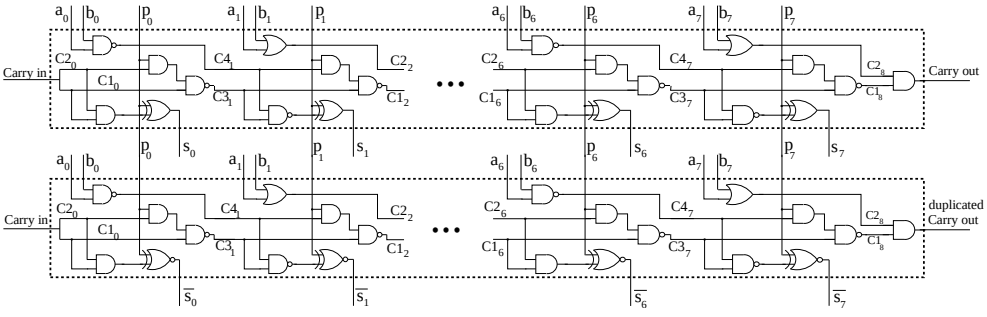

Figure 3: First sum-bit duplicated fast carry-ripple adder block

In the design of the block $SDB_1(8)$ the carry-in signals c_01 and c_02 are not specified. Usually the carry-in signals for the first block can be assumed to be zero. Then the block $SDB_1(8)$ can be easily simplified.

All the adder blocks $SDB_2^0(8)$, $SDB_2^1(8),\ldots,$ $SDB_6^0(12)$, $SDB_6^1(12)$ are very similar to the described adder block $SDB_1(8)$.

3 Self-checking property of the proposed carry-select adder

Now we show that the proposed sum-bit duplicated carry-select adder which is represented in Fig. 1 is code-disjoint and totally self-checking with respect to all single stuck-at faults. All errors due to single stuck-at faults are immediately detected if they for the first time occur. If a single (or an odd) error occurs at the inputs $a_0,\ldots,a_{63},p_a$ or $b_0,\ldots,b_{63},p_b$ or within the XOR-elements of the "*Propagate Generator*" the error will be detected by comparing $p(a\oplus b)$ with $p_a\oplus p_b$ and the adder is code-disjoint.

All errors due to single stuck-at faults of the gates which are connected either to the outputs $s_0,\ldots,s_{63}$ or $\overline{s}_0,\ldots,\overline{s}_{63}$ and all errors arising from faults in the multiplexors $MUXs_2$, $MUX\overline{s}_2,\ldots,MUXs_6$, $MUX\overline{s}_6$ for sum selection of the adder blocks are at once detected by comparing s and $\overline{s}$. If a fault of a gate changes only one of the carry-out values $c_j^0 1$, $c_j^0 2$, $c_j^1 1$ or $c_j^1 2$ for $j\in\{7,15,27,39,51\}$ then this erroneous carry-out signal is an erroneous select signal for sum selection and it will be also at once detected by comparing s and $\overline{s}$ in the succeeding adder block. Similarly faults in the multiplexors $MUXc_{15}1$, $MUXc_{15}2$,..., $MUXc_{51}1$,$MUXc_{51}2$ will result in erroneous control signals for the multiplexors for sum selection. Therefore they will be also immediately detected by comparing s and $\overline{s}$. It can be shown that for every single stuck-at fault there exists a test input for the proposed adder. Thus the proposed sum-bit duplicated adder is completely self-checking and also code-disjoint.

4 Experimental Results

The considered adders are modeled as a syntheticizable RTL description in $VHDL$ The library *lsi_10k* and the wire load model "10×10" were used. The adders and are modeled and optimized by the *SYNOPSYS CAD* tool of the *EUROCHIP* project. The adder cells of the proposed carry-select adder are described as a syntheticizable RTL description and only mapped to the library *lsi_10k*. Thus only a standard library is used.

The experimental results for 64 bit carry-select adders are shown in Table 1 and Table 2. The proposed sum-bit duplicated carry-select adder is compared with an ordinary carry-select adder without error detection and with a code-disjoint duplicated carry-select adder. All the considered adders have the same maximal delay for the most significant sum-bit. The area of the proposed adder is about 170% of the adder without error detection and about 79 % of the code-disjoint duplicated carry-select adder.

Table 1: Comparison of the proposed adder **vs.** adder without error detection

	Delay of s, ns(%)	Area, $unit$(%)
Adder without error detection	15 (100%)	2083 (100%)
Code-Disjoint Sum-bit Duplicated	15 (100%)	3559 (171%)

Table 2: Comparison of the proposed adder **vs.** duplicated adder

	Delay of s, ns(%)	Area, $unit$(%)
Code-Disjoint Duplicated	15 (100%)	4518 (100%)
Code-Disjoint Sum-bit Duplicated	15 (100%)	3559 (79%)

5 Conclusions

In this paper the first code-disjoint totally self-checking sum-bit duplicated carry-select adder was described. As adder blocks fast carry-ripple adders with a single NAND-gate delay for the carry-propagation per adder cell were used. The input parity is checked against the XOR-sum of the internal propagate signals. The parity checked propagate signals were utilized to implement both the duplicated adder blocks with the constant carry-in signals 0 and 1 which are inherent to a carry-select adder structure. It was demonstrated how it is possible effectively to implement for all the adder blocks both the corresponding sum-bits and their inverted values for concurrent checking. The adder is totally self-checking. For 64 bits for the SYNOPSYS CAD tool of the EUROCHIP project the block sizes 8,8,12,12,12 and 12 are experimentally found to be optimal. For 64 bits area and maximal delay are determined and compared with the results for a 64 bit carry-select adder without error detection. Compared to a carry-select adder without error detection the maximal delay for the most significant sum-bit does not increase and the additional area is growing only by about 71%. Compared to a completely duplicated code-disjoint carry-select adder we save 240 XOR-gates.

References

[BNZ96] Bedder, H., Nicolaidis, M., und Zorian, Y.: Achieving High Reliability in Low Cost Parity Prediction Array Arithmetic Operators. In: *Proceedings of 13th IEEE European Test Workshop.* Monpellier. 1996.

[KL03] Kumar, B. K. und Lala, P. K.: On-line Detection of Faults in Carry-Select Adders. In: *International Test Conference (ITC).* S. 912–918. 2003.

[Ni93] Nicolaidis, M.: Efficient Implementations of Self-checking Adders and ALU's. *FTSC 23.* S. 586–595. 1993.

[OGSM03] Ocheretnij, V., Gössel, M., Sogomonyan, E. S., und Marienfeld, D.: A Modulo *p* Checked Self-Checking Carry Select Adder. In: *9th International On-Line Testing Symposium.* S. 25–29. 2003.

[Pa00] Parhami, B.: *Computer Arithmetic. Algorithms and Hardware Designs.* Oxford University Press. 2000.

[Sh91] Shih, F.-H. W.: High performance self-checking adder having small circuit area. In: *US PS 5,018,093.* 1991.

[SHB68] Sellers(Jr.), F. F., Hsiao, M. Y., und Bearnson, L. W.: *Error Detection Logic for Digital Computers*. McGraw-Hill. 1968.

[Sm97] Smith, M.: *Application-specific integrated circuits*. Adison Wesley, Reading, MA. 1997.

[SOG01] Sogomonyan, E. S., Ocheretnij, V., und Gössel, M.: A new code-disjoint sum-bit duplicated carry look-ahead adder for parity codes. In: *10th Asian Test Symposium.* S. 365–370. 2001.