On the Design of Self-Testing Checkers for Modified Berger Codes

Stanisław J. Piestrak
Wrocław University of Technology
Institute of Engineering Cybernetics
50–370 Wrocław, Poland
piestrak@ict.pwr.wroc.pl

Dimitris Bakalis & Xrysovalantis Kavousianos
University of Patras & Computer Technology Inst.
Dept. of Computer Eng. & Informatics
26500 Patras, Greece
{bakalis,kabousia}@cti.gr

Abstract

One of several approaches for designing highly-reliable systems relies on using error detecting codes (EDCs) and implementing digital circuits as self-checking. One class of EDCs that has been very often used to implement self-checking circuits are Berger codes. Although several self-testing checkers (STCs) for Berger codes have been proposed in the past, they mostly present area and delay results based on gate counts and gate levels and not on real implementations. In this work we consider real implementations and present and evaluate the area, delay and power characteristics of STCs for modified Berger codes that are based on: (a) parallel counters and (b) sorting networks. Preliminary results indicate that STCs based on parallel counters are smaller and consume less power than the STCs based on sorting networks.

1 Introduction

This project is motivated by the growing needs to provide aerospace and life support systems (such as pacemakers) demanding both high-safety and low-power performance. In 1997 it has been reported that not only charged particles from the Sun but also neutrons have sufficient energy to flip bits in memories and corrupt logic inside processors [1]. The major concern is logic, which, unlike memory cannot be easily protected using parity checking or Hamming error correcting codes. The implementation of radiation hardened circuits is not always feasible or cost-efficient. The US Federal Aviation Administration confirmed that one in 10 avionic failures are unconfirmed on the ground, i.e. they might have been transient faults caused by cosmic radiation (called *single event upsets* (SEUs)).

One of several approaches for designing highly-reliable and/or safe systems relies on using error detecting codes (EDC's) and implementing digital circuits as self-checking (SC). One class of EDC's that has been very often used to implement SC circuits are Berger codes, which are the optimal systematic unordered codes capable of detecting unidirectional errors of any multiplicity. The reliable operation of a checker could be ensured by implementing it as a *self-testing checker* (STC) which is capable of detecting its own internal faults during normal functioning. Although several STCs for Berger and equivalent codes have been proposed in the past [2]–[10], they mostly present area and delay results based on gate counts and gate levels and not on real implementations. Actually, the only work that addressed the design of low-power consumption in SC circuits deals with 2-rail STCs only [11].

In this work we consider real implementations and present and evaluate the area, delay and power characteristics of STCs for modified Berger codes that are based on: (a) parallel counters and (b) sorting networks.

2 Preliminaries

2.1 Power Dissipation in CMOS VLSI Circuits

Two types of power dissipation in CMOS circuits are distinguished [12]: *dynamic* and *static*. Dynamic power dissipation is caused by *switching activities* of the circuits. Static power dissipation is related to the *logical states* of the circuits which, in CMOS logic, is caused by the leakage current only and it is generally negligible. The most significant source of dynamic power dissipation in CMOS circuits is the charging and discharging of capacitance. It strongly depends on the activity of the inputs and outputs of logic gates. Only the latter component of power dissipation will be taken into account here.

2.2 STCs and Berger Codes

Definition 1 *[13], [14] A circuit is called a* self-testing checker (STC) *if it is both:*

0-7695-1290-9/01 $10.00

1. Self-testing *(ST)*, i.e. *for every fault f from the set F of likely faults it produces a non-code output word for at least one input codeword; and*
2. Code-disjoint *(CD)*, i.e. *it maps the input code space to the output code space and the input non-code space to the output non-code space.*

Definition 2 *[15] The* Berger code $C_{(I,K)}$ *is a systematic code wherein the K check bits* $P_X = \{s_{K-1}, \ldots, s_1, s_0\}$ *are the binary representation of the bit-by-bit complemented number of 1s in the information part* $J_X = \{x_1, x_2, \ldots, x_I\}$, *where* $K = \lceil \log_2(I+1) \rceil$.

Definition 3 *[2] A* Maximal Length Berger *(MLB)* code $C_{MLB(K)}$ *is one for which* $K = \log_2(I+1)$, *i.e.* $I = 2^K - 1$.

One advantage of a MLB code is that all 2^K check parts are used, what facilitates the implementation of an STC. However, the most important class of the Berger codes are those with $I = 2^{K-1}$, i.e. $C_{(2^{K-1},K)}$. A combinational STC for Berger codes with $I = 2^{K-1}$ could not have been built until very recently [16]. At the time (1977), this limitation motivated the construction of a class of *equivalent modified Berger codes* for which a combinational STC could be built [2]. For $I = 2^{K-1}$ these codes are constructed as the concatenation of the MLB code with the 1-out-of-2 (1/2) code

$$C^*_{(2^{K-1},K)} = C_{(2^{K-1}-1,K-1)} \times C_{1/2}, \qquad (1)$$

where $C_{(2^{K-1}-1,K-1)}$ is the MLB code and two bits of a 1/2 codeword represent x_I — the most significant bit (MSB) of the information part, and s_{K-1} — the MSB of the check part, i.e. $s_{K-1} = \overline{x_I}$.

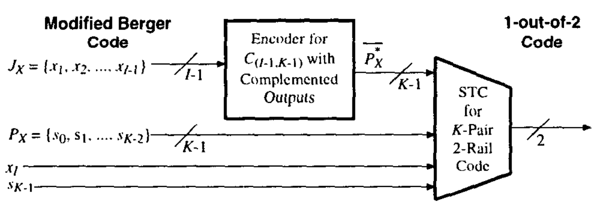

Figure 1. General structure of an STC for the modified Berger code $C^*_{2^{K-1},K}$ from [6].

Consequently, an STC for the modified Berger code $C^*_{(2^{K-1},K)}$ can be built as shown in Fig. 1 [6]. It consists of only two blocks:

- an encoder for the MLB code $C_{(2^{K-1}-1,K-1)}$ with complemented outputs, which is nothing else but the $(2^{K-1}-1)$-input counter of 1's; and
- an STC for the K-pair 2-rail code.

The latter circuit might have an arbitrary structure, since all 2^K combinations are used as check parts. One of the most efficient highly modular implementations is a tree of 2-pair 2-rail modules such as shown in Figure 7.

A counter of 1's can be built using two essentially different complementary concepts [7]:

1. as a *parallel counter (PC)* — a circuit built on the basis of carry-save adders (CSAs) entirely composed of full-adders (FAs) and half-adders (HAs); and
2. using a *multi-output threshold circuit* T^I followed by some NOT-AND-OR circuit.

The most hardware-efficient versions of the latter circuit, that will be referred to a *T^I-based counter of 1's*, can be designed by implementing T^I as a *sorting network (SN)* [6], [17]. Several realization of SNs can be found in [18]–[20]. Despite being the least complex, the SNs have also been shown easily-testable, since a circuit T^n implemented as a SN can be tested for all single stuck-at-z (s/z) faults, $z \in \{0,1\}$, using from $3n/2$ to $2n$ tests only [17], [21].

The gate count indicated that the SN-based STCs are superior than their PC-based counterparts for smaller I [7], [8]. Nevertheless, no realistic comparison of area, delay and power dissipation of these circuits has ever been reported.

3 Design of STCs for Modified Berger Codes with $I = 2^{K-1}$

We have shown in [22] that the 2-output combinational checker for Berger codes with $I = 2^{K-1}$ information bits proposed by Rao *et al.* [23] is not self-testing, as claimed. Essentially, such a checker can be built using a special STC for 2-rail codes, proposed recently in [16]. However, here we will consider a slightly simpler and faster STC for the modified Berger codes from [2], according to Fig. 1.

3.1 Design of Basic Blocks

Here we shall consider implementations of the STCs for the following modified Berger codes: $C^*_{(8,4)}$, $C^*_{(16,5)}$, $C^*_{(32,6)}$, and $C^*_{(64,7)}$.

The PC-based checkers employ the 7-, 15-, 31-, and 63-input PCs respectively built of 4, 11, 26, and 57 FAs. The SN-based checkers employ the circuits T^7, T^{15}, T^{31}, and T^{63} designed as follows. The circuit T^7 is the optimal SN by Batcher [18]. The circuit T^{15} is obtained by removing the bottom line along with all connected comparators in the optimal 16-input SN from [19] (see Fig. 2). The circuit T^{31} is built using the circuits T^{15} and T^{16} (either built according to [19]) followed by the 16-by-15 odd-even merging network (MN) designed according to Batcher [18]. A similar approach is used to build the circuit T^{32}. The latter is a part of the circuit T^{63}, which consists of the circuits T^{32} and T^{31} followed by the 32-by-31 odd-even MN. Either version of a checker employs an identical 4-, 5-, 6- or 7-pair TSC comparator built using 2-pair 2-rail modules (see Fig. 7).

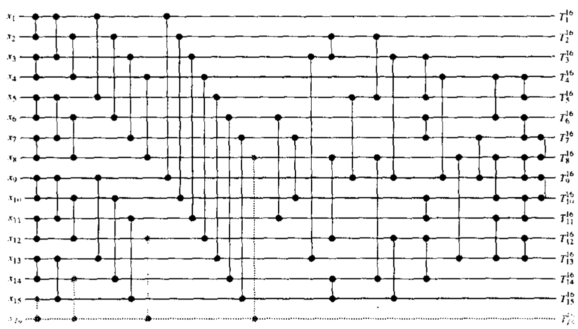

Figure 2. Optimal 16-input SN from [19].

3.2 Example: STCs for the Modified Berger Code with $I = 16$

To present the basic design ideas, we show the detailed structures of two versions of an STC for the modified Berger code $C^*_{(16,5)}$ in Figures 3 and 5. The logic schemes of all subcircuits needed to build these two realizations are shown in Figures 2–7.

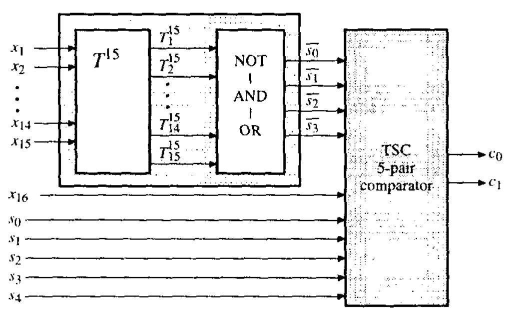

Figure 3. Internal structure of the T^{15} -based STC for the modified Berger code $C_{(16,5)}$.

Any STC for the modified Berger code $C^*_{(16,5)}$ has:

- 21 inputs: $I = 16$ information bits $\{x_1, x_2, \ldots, x_{16}\}$ and $K = 5$ check bits $\{s_4, s_3, s_2, s_1, s_0\}$ (s_4 is the most significant bit such that $s_4 = \overline{x_{16}}$); and
- two outputs $\{s_1, s_0\}$, which are complement of each other, when there are no input errors or internal faults.

$w(X)$	$\overline{s_3}$	$\overline{s_2}$	$\overline{s_1}$	$\overline{s_0}$
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

$$\overline{s_3} = T_8^{15}$$

$$\overline{s_2} = T_4^{15}\overline{T_8^{15}} + T_{12}^{15}$$

$$\overline{s_1} = T_2^{15}\overline{T_4^{15}} + T_6^{15}\overline{T_8^{15}} + T_{10}^{15}\overline{T_{12}^{15}} + T_{14}^{15}$$

$$\overline{s_0} = T_1^{15}\overline{T_2^{15}} + T_3^{15}\overline{T_4^{15}} + T_5^{15}\overline{T_6^{15}} + T_7^{15}\overline{T_8^{15}} + T_9^{15}\overline{T_{10}^{15}} + T_{11}^{15}\overline{T_{12}^{15}} + T_{13}^{15}\overline{T_{14}^{15}} + T_{15}^{15}$$

Figure 4. Truth table and logic functions of the 15-input counter of 1's with complemented outputs, built using T^{15}.

The principal part of the T^{15}-based STC for the modified Berger code $C^*_{(16,5)}$ designed according to [6] (see Fig. 3) is the T^{15}-based counter of 1's. It implements the functions derived with the help of the truth table, both shown in Fig. 4. It is a circuit composed of the 15-input SN (see Fig. 2) with 15-inputs $\{x_1, x_2, \ldots, x_{15}\}$ and 15 outputs $\{T_1^{15}, T_2^{15}, \ldots, T_{15}^{15}\}$ followed by some NOT-AND-OR circuit with four outputs $\{\overline{s_3}, \overline{s_2}, \overline{s_1}, \overline{s_0}\}$, which are the regenerated complemented check bits. These are then compared against the set of original check bits $\{s_3, s_2, s_1, s_0\}$ along with a pair of bits s_4 and x_{16}.

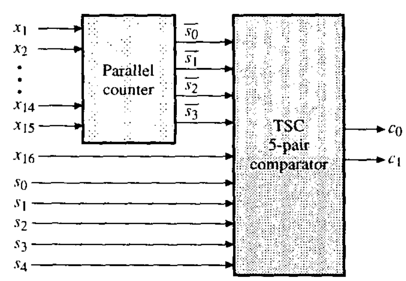

Figure 5. Internal structure of the PC-based STC for the modified Berger code $C_{(16,5)}$.

The PC-based counter of 1's also has $I = 15$ information bits $\{x_1, x_2, \ldots, x_{15}\}$ as inputs and, similarly to its T^{15}-based counterpart, it generates $K = 4$ complemented check bits directly $\{\overline{s_3}, \overline{s_2}, \overline{s_1}, \overline{s_0}\}$ (see Fig. 6). The remaining part of a circuit works in the same way as for the T^{15}-based STC.

As for the gate-level complexity of the two realizations considered, note the following. Assuming that: (i) a FA is built of 10 gates and (ii) the least complex known 15-input threshold circuit (obtained by modifying the least complex known 16-input SN from [20]) built of 56 comparators (i.e. 112 2-input gates) in 15 levels are used. The overall complexity of the T^{15}-based counter of 1's is 133 gates compared to 110 gates of the PC-based counter.

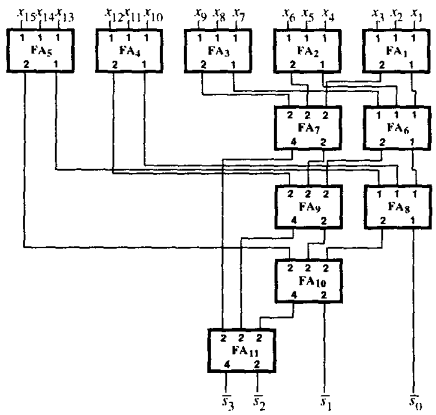

Figure 6. 15-input parallel counter.

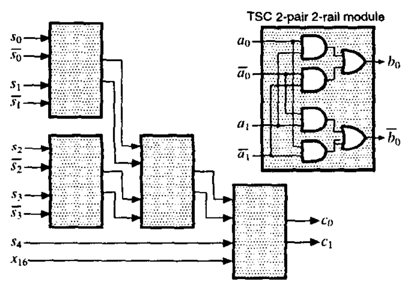

Figure 7. 5-pair TSC comparator.

The 5-pair 2-rail TSC comparator, which could be built using four identical TSC 2-pair 2-rail modules as shown in Fig. 7, is identical for either version of a counter of 1's.

4 Experimental Results

We have described in structural HDL language PC-based and SN-based STCs for the following modified Berger codes: $C^*_{(8,4)}$, $C^*_{(16,5)}$, $C^*_{(32,6)}$, and $C^*_{(64,7)}$. We use the Design Analyzer tool by Synopsys to map each one of the above circuits to the 0.6μm CMOS VLSI implementation technology and then proceed to optimizations. At first we target speed. We repetitively synthesize each one of the circuits until no faster design can be obtained. We then recover as much area as possible without affecting the delay. Table 1 presents the area and delay characteristics of the delay-optimized circuits. We have to note that the area estimations include the routing area. From Table 1 we can observe that when I is small ($I \leq 16$), the SN-based circuit is slightly faster than its PC-based counterpart. However, for larger values of I ($I > 16$) the PC-based circuits are faster. It is also obvious that all SN-based circuits require much more area than the PC-based circuits. This is mainly because the SN-based circuits require much more area for their interconnections (approximately 75% of the total area is consumed by the interconnections).

Table 1. Delay-optimized designs.

Circuit		Area [mm^2]	Delay [ns]
$C^*_{(8,4)}$	PC-based	99	2.56
	SN-based	140	2.26
$C^*_{(16,5)}$	PC-based	175	4.29
	SN-based	442	4.17
$C^*_{(32,6)}$	PC-based	395	6.65
	SN-based	1022	7.00
$C^*_{(64,7)}$	PC-based	611	8.18
	SN-based	2294	8.75

Table 2. Area-optimized designs.

Circuit		Area [mm^2]	Delay [ns]
$C^*_{(8,4)}$	PC-based	56	4.26
	SN-based	75	4.46
$C^*_{(16,5)}$	PC-based	113	6.93
	SN-based	222	6.11
$C^*_{(32,6)}$	PC-based	231	9.36
	SN-based	616	10.85
$C^*_{(64,7)}$	PC-based	483	10.41
	SN-based	1700	13.11

Table 3. Power estimation.

Circuit		Area [mm^2]	Delay [ns]	f [MHz]	Power [mW]
$C^*_{(8,4)}$	PC-based	69	2.97	333	15.37
	SN-based	117	2.99	333	22.95
$C^*_{(16,5)}$	PC-based	167	5.00	200	27.81
	SN-based	316	5.00	200	33.39
$C^*_{(32,6)}$	PC-based	285	7.49	133	31.67
	SN-based	765	7.49	133	43.94
$C^*_{(64,7)}$	PC-based	509	9.89	100	42.64
	SN-based	2287	10.00	100	81.34

Since there are several applications where area is more critical than delay, we repeated the synthesis process targeting low area. From Table 2 one can derive similar observations as before: the PC-based designs are not only smaller than the SN-based designs but are also faster for larger I.

For comparing the power dissipation of two circuits, one must assume that they have the same clock frequency f. Hence, the two circuits should be optimized in such a way that they have approximately the same delay which corresponds to that frequency (further delay optimizations would be meaningless since the operating clock frequency is fixed). To this end, assuming a system clock of 333, 200, 133 and 100 MHz for the circuits with $I = 8$, 16, 32, and 64 respectively, we have optimized the SN-based and PC-based circuits so as their delays to be less than 3.0, 5.0, 7.5 and 10.0 ns, respectively, and proceeded to power estimation. The results regarding area, delay and power dissipation are given in Table 3. It is evident that the SN-based circuits consume much more power than the equivalent PC-based circuits.

5 Conclusions

We have considered real implementations of the self-testing checkers (STCs) for modified Berger codes with $I = 2^{K-1}$ information bits. The simulation results of dynamic power dissipation for the two versions performed using power estimation CAD tools by Synopsys provided very interesting results. They revealed that the PC-based structures are more advantageous for designing STCs, since they are smaller, faster (when the size of the information part is medium or large), and consume less power than the structures built using sorting networks.

6 Acknowledgement

The authors would like to thank Prof. D. Nikolos for the fruitfull discussions concerning the presented work.

References

[1] D. Larner, "Sun flips bits in chips," *Electronics Times*, No. 878, p. 72 and 24, 10 Nov. 1997.

[2] M. J. Ashjaee and S. M. Reddy, "On totally self-checking checkers for separable codes," *IEEE Trans. Comput.*, vol. C-26, pp. 737–744, Aug. 1977.

[3] M. A. Marouf and A. D. Friedman, "Design of self-checking checkers for Berger codes," in *Dig., Pap. 8th Int. FTC Symp.*, Toulouse, France, June 1978, pp. 179–184.

[4] S. J. Piestrak, "Design of fast self-testing checkers for Berger codes," *IEEE Trans. Comput.*, vol. C-36, pp. 629–634, May 1987.

[5] J.-Ch. Lo and S. Thanawastien, "The design of fast totally self-checking Berger code checkers," in *Dig., Pap. 18th Int. FTC Symp.*, Tokyo, Japan, June 1988, pp. 226–231.

[6] S. J. Piestrak, "The minimal test set for sorting networks and the use of sorting networks in self-testing checkers for unordered codes," in *Dig., Pap. 20th Int. FTC Symp.*, Newcastle upon Tyne, UK, June 26–28, 1990, pp. 457–464.

[7] —, *Design of Self-Testing Checkers for Unidirectional Error Detecting Codes*, Scientific Papers of the Inst. of Techn. Cybern. of the Techn. Univ. of Wrocław, No. 92, Ser.: Monographs No. 24, Oficyna Wyd. Polit. Wrocł., Wrocław 1995, 112 pp.

[8] —, "Design of encoders and self-testing checkers for some systematic unidirectional error detecting codes," *Int. J. on Microelectronic Systems Integration*, vol. 5, pp. 246–260, No. 4, 1997.

[9] Yu-Yau Guo, Jien-Chung Lo, and Metra C., "Fast and area-time efficient Berger code checkers," *Proc. 1997 IEEE Int. Symp. on DFT in VLSI Systems*, pp. 110–118.

[10] X. Kavousianos, D. Nikolos, G. Foukarakis, and T. Gnardellis, "New efficient totally self-checking Berger code checkers," *Integration, The VLSI Journal*, vol. 28, Issue 1, pp. 101-118, 1 Sep. 1999.

[11] M. Favalli and C. Metra, "Design of low-power CMOS two-rail checkers," *Int. J. on Microelectronic Systems Integration*, vol. 5, pp. 101–110, No. 2, 1997.

[12] G. K. Yeap, *Practical Low Power Digital VLSI Design*, Kluwer Academic Publishers, Boston, 1997.

[13] W. C. Carter and P. R. Schneider, "Design of dynamically checked computers," in *Proc. IFIP Conf.*, Edinburgh, Scotland, Aug. 1968, pp. 878–883.

[14] D. A. Anderson and G. Metze, "Design of totally self-checking check circuits for m-out-of-n codes," *IEEE Trans. Comput.*, vol. C-22, pp. 263–269, Mar. 1973.

[15] J. M. Berger, "A note on error detection codes for asymmetric binary channels," *Inform. Contr.*, vol. 4, pp. 68–73, Mar. 1961.

[16] S. J. Piestrak, "Design method of a class of embedded combinational self-testing checkers for 2-rail codes," *IEEE Trans. Comput.*, vol. C-50, 2001.

[17] —, "The minimal test set for multi-output threshold circuits implemented as sorting networks," *IEEE Trans. Comput.*, vol. 42, pp. 700–712, June 1993.

[18] K. E. Batcher, "Sorting networks and their applications," in *Proc. 1968 SJCC — Spring Joint Computer Conf.*, AFIPS, vol. 32, 1968, pp. 307–314.

[19] D. E. Knuth, *The Art of Computer Programming, Vol. III, Sorting and Searching*, 2nd Ed., Reading, MA, Addison-Wesley, 1973, ch. 5.

[20] D. C. Van Voorhis, "An economical construction for sorting networks," in *Proc. AFIPS NCC — Nat. Computer Conf.*, 1974, pp. 921–927.

[21] S. J. Piestrak and A. Dandache, "Minimal test set for multi-output threshold circuits implemented as bubble sorting networks," *Electr. Lett.*, vol. 36, pp. 202–204, No. 3, 2000.

[22] S. J. Piestrak, "Comments on 'Novel totally self-checking Berger checker designs based on generalized Berger code partitioning'," subm. in Jan. 1999 to *IEEE Trans. Comput.*

[23] T. R. N. Rao *et al.*, "Novel totally self-checking Berger checker designs based on generalized Berger code partitioning," *IEEE Trans. Comput.*, vol. 42, pp. 1020–1024, Aug. 1993.