JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 15, 429-441 (1999)
DOI:10.1688/JISE.1999.15.3.8

Short Paper

TSC Berger-Code Checker Design for 2^{r-1}-Bit Information

WEN-FENG CHANG AND CHENG-WEN WU*
Department of Computer Science
**Department of Electrical Engineering*
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.

Berger code (BC) is an optimal separable code which can detect all single stuck-at and unidirectional faults. The traditional BC checker is not totally self-checking (TSC) when $k = 2^{r-1}$, where k is the number of information bits in a code word, and r is the number of check bits. The case in which $k = 2^{r-1}$, however, is important and practical since information words in most digital systems are in the form of a power of 2. This problem was solved recently by Rao *et al*., who proposed a TSC BC checker based on the concept of a generalized BC partition technique. We extend their work and propose two TSC BC checkers for $k = 2^{r-1}$ by modifying the conventional BC checker. The first design adds to it a pair of 1/2-code inputs and a two-rail-code checker. The second method includes a periodic input and produces a periodic output. Compared with the results of Rao *et al*., our checkers require less hardare overhead and check time.

Keywords: Berger code, concurrent error detection, on-line testing, totally-self-checking checker, two-rail code

1. INTRODUCTION

The totally self-checking (TSC) technique is important in digital systems which require high reliability and dependability, such as the computer system on an aircraft, the navigation system on a spacecraft, etc. It can detect both temporary and permanent faults on line (i.e., concurrently with the system in its normal routine) [1, 2]. Therefore, software diagnostic programs can be removed or reduced without losing system reliability/dependability [2].

Concurrent error detection for arithmetic and logic circuits has long been identified as an important issue in designing highly reliable and dependable computing machines (see, e.g., [3-13]). Error control coding has been shown to be effective for this purpose. One of the most important classes of codes for arithmetic and logic circuits is that of the *Berger codes* (BCs) [4], which are systematic codes. A BC of length n, denoted as BC(n, k), has k information bits (which form the information symbol I) and $r = n - k$ check bits (which form the check symbol I_c), where I_c is the binary number representing the number of zero bits in I, and $|I_c| = r = \lceil \log_2(k+1) \rceil$. It has been shown that BCs are optimal systematic

Received October 7, 1997; revised April 21, 1998; accepted June 8, 1998.
Communicated by Chi Sung Laih.

AUED codes which form the basis of Bose-Lin codes [8, 14, 15]. They have been used frequently to handle arithmetic and logic operations [10, 11, 15, 16]. Designing low-cost BC checkers, therefore, is an important task in realizing dependable digital systems.

Let the input code space be X, and the output code space be Y. A normal circuit (checker) will map each $x \in X$ to a $y \in Y$. If a fault f exists, then the TSC conditions defined as follows will help us detect the fault [1]. A circuit C is *fault-secure* (FS) for a set of faults F if $\forall f \in F$, $\forall x \in X$, either $C_f(x) \notin Y$ or $C_f(x) = C(x)$, where $C_f(x)$ represents the faulty circuit (function). A circuit C is *self-testing* (ST) for a set of faults F if $\forall f \in F$, $\exists x \in X$ such that $C_f(x) \notin Y$. A circuit C is *totally self-checking* (TSC) if it is both FS and ST. A circuit C is *code-disjoint* (CD) if $\forall x \in X$, $C(x) \in Y$, and $\forall \hat{x} \notin X$, $C(\hat{x}) \notin Y$. A circuit is a *TSC checker* if it is both TSC and CD.

The general form of a Berger-code TSC checker [17, 18](called the *general BC checker*) is shown in Fig. 1. Note that the check bits (I_c) from the functional circuit and the outputs of the Complementary Check-Bit Generator G should be bit-wise complementary to each other when the system is fault-free, and that there should be at least one pair of identical bits from the respective bit positions of I_c and $\bar{I}_c$, which can be guaranteed by a subsequent two-rail-code (TRC) checker tree. However, the general BC checker is not TSC when $k = 2^{r-1}$ because in this case, the TRC tree is not fully tested by its input code words (i.e., it is not ST). The case where $k = 2^{r-1}$ is important and practical since information words in most digital systems are in the form of a power of 2.

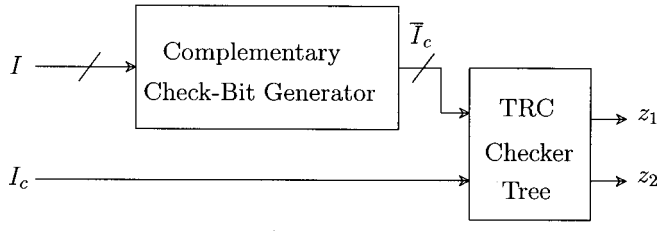

Fig. 1. The general TSC berger-code checker.

This problem was solved recently by Rao *et al*. [19, 20], who proposed a TSC BC checker based on the concept of a generalized BC partition technique. (The original BC partition technique was proposed by Piestrak [18].) Fig. 2 shows the block diagram of their TSC checker for BC(n, k), where $k = 2^{r-1}$, $r = n - k$ [19, 20]. The subcircuits W_1 and W_2 implement the $\Sigma x_i + \delta_a$ and $I_c + \delta_b$ functions, respectively, where $\delta_a + \delta_b = 2^r - 1 - k$. The outputs of W_1 and W_2 are bit-wise complementary to each other and are sent to a TRC checker tree. Note that every TRC checker in this case receives four different patterns [19, 20], so the ST condition is satisfied. This checker, however, requires adders for W_1 and W_2, whose complexity is high.

In this paper, we extend their work and propose two TSC BC checkers for the same case, i.e., BC(n, k), where $k = 2^{r-1}$, $r = n - k$. As in [19, 20], our fault model is the single stuck-at fault. We use a different approach: our checkers are based on the general Berger-

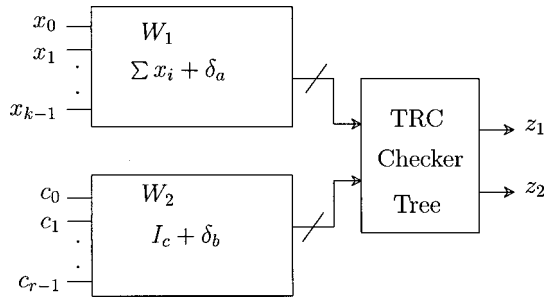

Fig. 2. A TSC berger-code checker for BC(n, k), where $k = 2^{r-1}$, $r = n - k$.

code checker for $k \neq 2^{r-1}$ as shown in Fig. 1. In the first design, BC1, a pair of 1/2-code inputs is required in addition to the TRC checker tree; i.e., the ST condition is satisfied with the help of external 1/2-code inputs instead of adder circuits. The external 1/2-code inputs can come from another checker in the system or from external complementary signals. The second method, BC2, uses a periodic input and produces a periodic output when the system is operating normally. If the system is faulty, then the output becomes aperiodic. BC1 is suitable for a conventional TSC system while BC2 can be used in a system where a clock signal can be used as the periodic input. Both BC1 and BC2 are cheaper and faster compared with the ones reported in [19, 20], although they have the above input restrictions.

2. TSC BC CHECKERS

2.1 BC1

BC1, shown in Fig. 3, is composed of three subcircuits:

1. For the $\sum_{i=0}^{k-1} \bar{x}_i$ counter (0's counter), the outputs y_{r-1}, y_{r-2}, ..., y_1, y_0 are the binary representation of the number of zero bits in the inputs. The counter outputs are complementary.
2. The TRC-TREE has $r - 1$ pairs of two-rail inputs (i.e., $(\bar{y}_0, c_0), (\bar{y}_1, c_1), \ldots, (\bar{y}_{r-2}, c_{r-2})$) as shown in Fig. 3(b) in a recursive form: it is composed of two smaller TRC-TREE's (TT-1 and TT-2) with $\lfloor (r-1)/2 \rfloor$ and $\lceil (r-1)/2 \rceil$ input pairs, respectively, and a two-variable TRC checker (shown as TRC in the figure) which compares the outputs of TT-1 and TT-2. Under fault-free operation, it will receive an exhaustive number of code inputs (see Example 1). The two-variable TRC checker is shown in Fig. 4(a). Under fault-free operation, it receives two input pairs, (a_i, b_i) and (a_j, b_j), where $a_i = \bar{b}_i, a_j = \bar{b}_j$, and generates a 1/2-code output pair, (f, g). It can be tested by patterns 0101, 0110, 1001, and 1010.

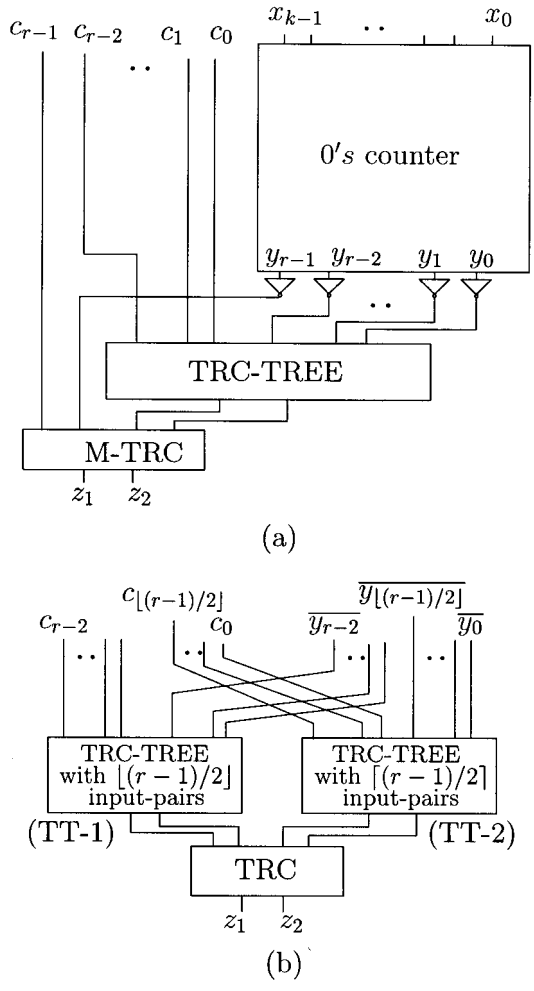

Fig. 3. (a) Our TSC checker for BC(n, k), where $k = 2^{r-1}$, $r = n - k$. (b) TRC–TREE.

3. The modified TRC checker (M-TRC in Fig. 3(a)), which is shown in detail in Fig. 4(b), is composed of two TRC checkers with a set of external 1/2-code inputs, (f_i, g_i), which are assumed to be provided by another TSC checker in the system or by external complementary signals. It receives three pairs of inputs, i.e., (c_{r-1},$\overline{y}_{r-1}$), (f_i, g_i), and the outputs from the TRC-TREE. Under fault-free operation, the inputs of M-TRC have only three possible combinations for $a_1b_1a_2b_2$: 0110, 1001, and 0101 (see Example 1). Its partial truth table is shown in Table 1. Note that both TRC 1 and TRC 2 receive all four possible input patterns, 1010, 1001, 0110, and 0101, even when there are only three different combinations for $a_1b_1a_2b_2$. Therefore, M-TRC is ST.

Table 1. Partial truth table for M-TRC.

a_1	b_1	a_2	b_2	f_i	g_i	t_1	t_2	z_1	z_2
0	1	1	0	1	0	1	0	1	0
0	1	1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0
1	0	1	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1

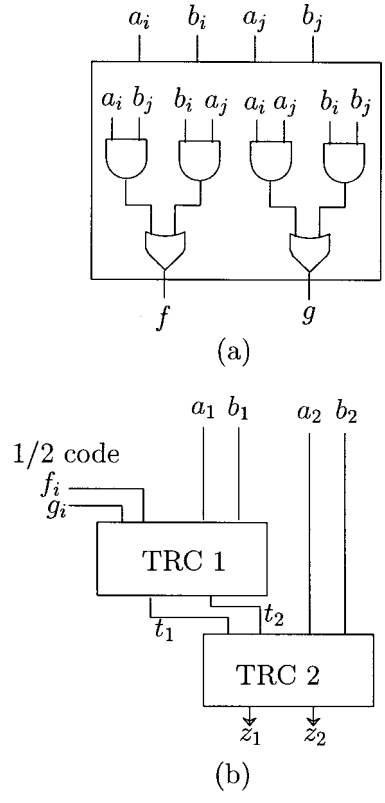

Fig. 4. (a) A two-variable TRC checker. (b) A modified TRC (M-TRC) checker.

Table 2. Possible combinations of $y_3y_2y_1y_0$ in Fig. 5(b).

$\sum_{i=0}^{7} \bar{x}_i$	$y_3y_2y_1y_0$
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

Example 1: The complementary full-adder (C-FA, whose function is $2C+S = \bar{a}+\bar{b}+\bar{c}$) and the complementary half-adder (C-HA, whose function is $2C+S = \bar{a}+\bar{b}$), which count the number of 0's in their inputs, are shown in Fig. 5(a). Our TSC checker for BC(12, 8) is shown in Fig. 5(b). The 0's counter is ST because it is non-redundant and can receive all 2^8 input combinations. The 9 possible output combinations of $y_3y_2y_1y_0$ for the counter are listed in Table 2, in which it can be seen that $y_2y_1y_0$ are assigned all $2^3 = 8$ combinations, and $c_i = y_i$, $0 \le i \le 2$, under fault-free operation. It allows TRC-TREE to receive all two-rail code words. Therefore, TRC-TREE is ST. As to M-TRC, its inputs $a_1b_1a_2b_2$ have only three possible combinations since y_3 produces a 1 in only one case out of the nine possible outputs from the counter (see Table 2). This only occurs when $c_3c_2c_1c_0x_7x_6\ldots\ x_1x_0$ = 100000000000, i.e., the information bits are all zero, and M-TRC receives $a_1b_1a_2b_2$ = 1001. Note that $a_1b_1a_2b_2$ = 0101/0110 can be received by M-TRC when $y_3y_2y_1y_0$ = 0000/0001. However, it is sufficient to test M-TRC even when it can receive only 1001, 0101, and 0110. The discussion is true for any BC(n, k), where $k = 2^{r-1}$, $r = n - k$, because only the code input $c_{r-1}c_{r-2}\ldots c_1c_0x_{k-1}\ldots x_1x_0$ = 100...00 can result in $y_{r-1} = 1$. □

Proposition 1: BC1 is a TSC BC(n, k) checker, where $k = 2^{r-1}$, $r = n - k$.

Proof: It is easy to see that BC1 is CD and FS. The 0's counter is ST because it is nonredundant and receives exhaustive inputs. The TRC-TREE is ST because it is non-redundant and receives all two-rail code inputs. Finally, M-TRC is ST because it receives all 3 patterns it needs as discussed above. Therefore, BC1 is a TSC BC(n, k) checker for $k = 2^{r-1}$ and $r = n - k$. □

2.2 BC2

BC2 is similar to BC1 (Fig. 3) except that M-TRC is replaced by M-TRC* (Fig. 6), which is a two-variable TRC checker proposed in [21]. It is composed of four XOR gates and a special CMOS gate (CMOS-GATE* in Fig. 6) with a periodically changing input signal s. In normal operation, it receives two pairs of two-rail code inputs, (a_1, b_1) and (a_2, b_2), and produces a periodically changing output Z ($Z = t_1 = t_2 = \bar{s}$). CMOS-GATE* can be

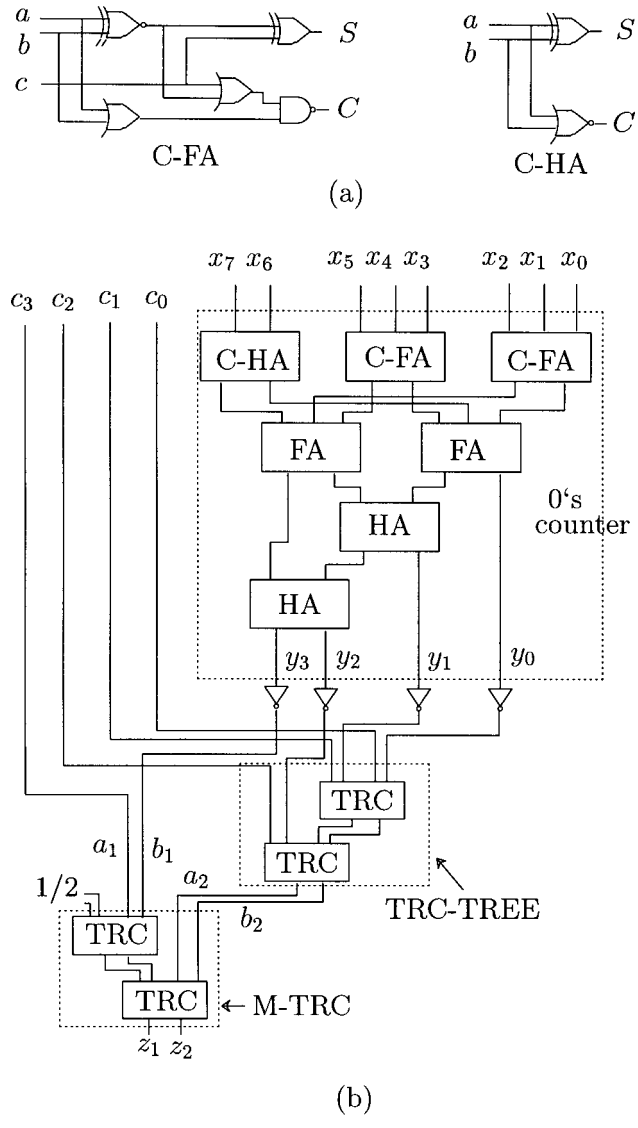

Fig. 5. (a) C-FA and C-HA. (b) The TSC checker (BC1) for BC(12, 8).

tested by patterns 00 and 11. By observing the output Z to see whether it is periodic, we can detect non-TRC input or stuck-at fault in M-TRC*. The partial truth table is shown in Table 3, from which we can see that each XOR gate can receive all its test patterns, 00, 01, 10 and 11, and CMOS-GATE* also can receive both its test patterns, 00 and 11, even when there are only three different combinations for $a_1b_1a_2b_2$. Therefore, M-TRC* is ST in BC2. Note that BC2 is considered to be faulty if Z is aperiodic.

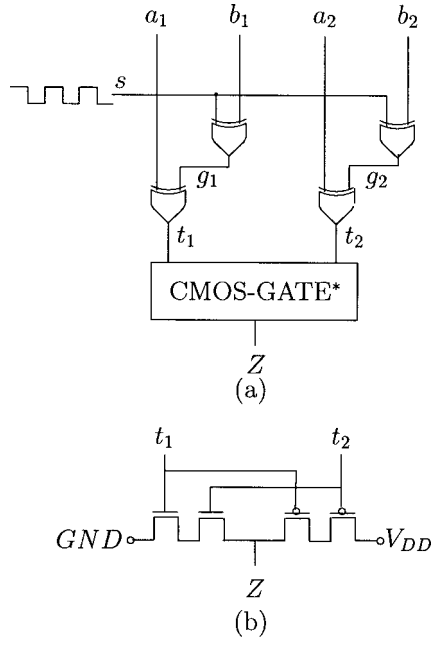

Fig. 6. (a) M-TRC*. (b) CMOS-GATE*.

Table 3. Partial truth table for M-TRC*.

a_1	b_1	a_2	b_2	s	g_1	g_2	t_1	t_2	Z
0	1	1	0	0	1	0	1	1	1
0	1	1	0	1	0	1	0	0	0
1	0	1	0	0	0	0	1	1	1
1	0	1	0	1	1	1	0	0	0
0	1	0	1	0	1	1	1	1	1
0	1	0	1	1	0	0	0	0	0

Example 2: BC2 for BC(12,8) is shown in Fig. 7. Note that it has only one primary output *Z*. By observing the output *Z* to see whether it is periodic, we can detect a noncode input or stuck-at fault in the checker itself. □

It is easy to see that BC2 is CD and FS. The discussion of ST of BC2 is similar to BC1.

Proposition 2: BC2 is a TSC BC(*n*, *k*)checker, where $k = 2^{r-1}$, $r = n - k$.

BC1 is suitable for conventional TSC systems but requires the existence of another checker in the system or a pair of external complementary input signals. BC2 can use the system clock as the added input, which works even when there is no other checker in the system.

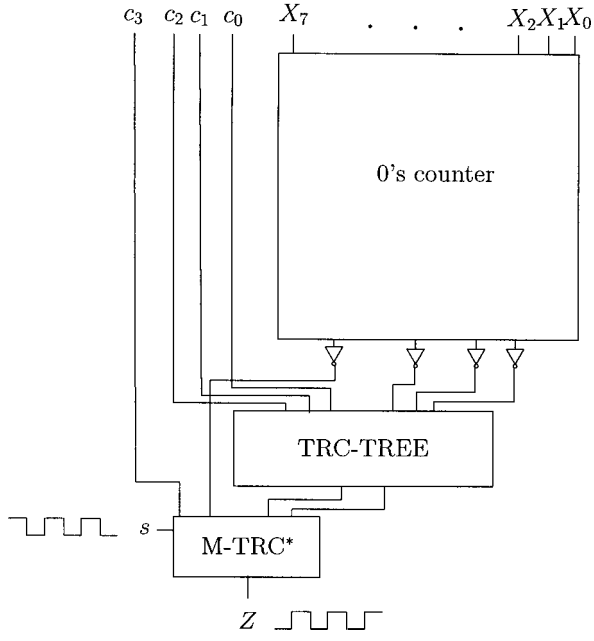

Fig. 7. BC2 for BC(12,8).

3. HARDWARE AND TIME COMPLEXITY

The 0's counter of our BC1 and BC2 for a BC(n, k) code, where $k = 2^{r-1}$ and $r = n - k$, requires $(k - r)$ FA/C-FA's and W[$2^r - 1 - k$] HA/C-HA's, where $r = \lfloor \log_2 k + 1 \rfloor$ and $W[x]$ denotes the Hamming weight of the binary number x. TRC-TREE requires $r - 2$ two-variable TRC checkers, and M-TRC requires two TRC checkers. M-TRC* requires 10 gate inputs, FA/C-FA requires 10 gate inputs, HA/C-HA requires 4 gate inputs, and the two-variable TRC checker requires 12 gate inputs (considering all external and internal gate inputs). Therefore, our TSC checkers BC1 and BC2 require GI_1 and GI_2 gate inputs, respectively:

$$\begin{aligned} GI_1 &= 10(k - r) + 4W[2^r - 1 - k] + 12r \\ &= 10k + 2r + 4W[2^r - 1 - k], \\ GI_2 &= 10(k - r) + 4W[2^r - 1 - k] + 12(r - 2) + 10 \\ &= 10k + 2r + 4W[2^r - 1 - k] - 14, \end{aligned}$$

where $r = \lfloor \log_2 k + 1 \rfloor$.

The previous checker proposed in [19, 20] (as shown in Fig. 2) requires (1) a subcircuit $\Sigma x_i + \delta_a$ (whose gate input count equals that of our 0's counter + 6), (2) a subcircuit $I_c + \delta_b$ which requires an $MHA_{(2)}$, $r - 3$ $MHA_{(1)}$'s, an HA, and an $MHA_{(3)}$, and 3) $r - 1$ TRC checkers, where $r = \lfloor \log_2 k + 1 \rfloor$. Each $MHA_{(1)}$ requires 10 gate inputs, an $MHA_{(2)}$ requires 1 gate input, and an $MHA_{(3)}$ requires 8 gate inputs. Therefore, the TSC BC checker in [19, 20] requires the following number of gate inputs:

$$GI_3 = 10(k - r) + 4W[2^r - 1 - k] + 6 + 1 \cdot 1$$
$$+10 \cdot (r - 3) + 4 \cdot 1 + 8 \cdot 1 + 12(r - 1)$$
$$= 10k + 12r + 4W[2^r - 1 - k] - 23,$$

where $r = \lfloor \log_2 k + 1 \rfloor$.

The time complexity of our checkers is $O(r)$ due to its tree structure. Comparisons of the hardware complexity and gate-delay between the checkers in [19, 20] and ours are shown in Tables 4 and 5, respectively. Table 4 shows that BC1 and BC2 require less hardware than do the checkers in [19, 20] by an amount of $10r - 23$ and $10r - 9$ gate inputs, respectively. Table 5 shows that BC1 and BC2 are faster than the checkers in [19, 20] by an amount of $2r - 8$ and $2r - 7$ gate delays, respectively.

Table 4. Comparison of the number of gate inputs.

k	r	[19]	GI_1	GI_2	[19] – GI_2
8	4	117	100	86	31
16	5	213	186	172	41
32	6	389	352	338	51
64	7	725	678	664	61
128	8	1381	1324	1310	71

Table 5. Comparison of gate delays.

k	r	[19]	GD_1	GD_2	[19] – GD_2
8	4	14	14	13	1
16	5	24	22	21	3
32	6	34	30	29	5
64	7	44	38	37	7
128	8	54	46	45	9

4. CONCLUSIONS

Two low-cost and fast TSC checkers for BC(n, k) for the case where $k = 2^{r-1}$ have been proposed. They are based on a conventional Berger-code checker for $k \neq 2^{r-1}$. The first design adds to it a pair of 1/2-code inputs and a two-rail-code (TRC) checker to satisfy the self-testing condition. The second method includes a periodic input and produces a periodic output. Our checkers require less hardware than do previously published checkers by an amount of $10r - 23$ and $10r - 9$ gate inputs, respectively, and are faster by an amount of $2r - 8$ and $2r - 7$ gate delays, respectively, although they have the above input restrictions.

REFERENCES

1. D. A. Anderson and G. Metze, "Design of totally self-checking check circuits for *m*-out-of-*n* codes," *IEEE Transactions on Computers*, Vol. 22, No. 3, 1973, pp. 263-269.
2. M. A. Marouf and A. D. Friedman, "Efficient design of self-checking checker for any *m*-out-of-*n* code," *IEEE Transactions on Computers*, Vol. 27, No. 6, 1978, pp. 482-490.
3. W. W. Peterson, "On checking an adder," *IBM Journal of Research and Development*, Vol. 2, April, 1958, pp. 166-168.
4. J. M. Berger, "A note on error detecting codes for asymmetric channels," *Information and Control*, Vol. 4, March, 1961, pp. 68-73.
5. A. Avizienis, "Arithmetic algorithms for error-coded operands," *IEEE Transactions on Computers*, Vol. 22, No. 6, 1973, pp. 567-572.
6. J. E. Smith, "The design of totally self-checking check circuits for a class of unordered codes," *Journal of Design Automation and Fault-Tolerant Computing*, Vol. 2, Oct. 1977, pp. 321-343.
7. J. B. Clary and R. A. Sacane, "Self testing computer," *IEEE Computer*, Vol. 12, No. 10, pp. 49-59.
8. B. Bose and D. J. Lin, "Systematic unidirectional error-detecting codes," *IEEE Transactions on Computers*, Vol. 34, No. 11, 1985, pp. 1026-1032.
9. A. M. Paschalis, D. Nikolos and C. Halatsis, "Efficient modular design of TSC checker for *m*-out-of-2*m* codes," *IEEE Transactions on Computers*, Vol. 37, No. 3, 1988, pp. 301-309.
10. J.-C. Lo, S. Thanawastien and T. R. N. Rao, "Concurrent error detection in arithmetic and logical operations using berger codes," in *Proceedings of International Test Conference (ITC)*, 1990, pp. 233-240.
11. J.-C. Lo, S. Thanawastien, T. R. N. Rao and M. Nicolaidis, "An SFS berger check prediction ALU and its applications to self-checking processor designs," *IEEE Transactions on Computer-Aided Design*, Vol. 11, No. 4, 1992, pp. 525-540.
12. M. Nicolaidis, "Efficient implementations of self-checking adders and ALUs," in *Proceedings of International Symposium on Fault Tolerant Computing (FTCS)*, 1993, pp. 586-595.
13. M. Nicolaidis and H. Bederr, "Efficient implementations of self-checking multiply and divide arrays," in *Proceedings of European Design and Test Conference (EDAC-ETC-Euro ASIC)*, (Paris), 1994, pp. 574-579.
14. C. V. Friedman, "Optimal error detecting codes for asymmetric binary channels," *Information and Control*, Vol. 5, No. 3, 1962, pp. 64-71.
15. T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
16. J.-C. Lo, S. Thanawastien and T. R. Rao, "Berger check prediction for array multipliers and array dividers," *IEEE Transactions on Computers*, Vol. 42, No. 7, 1993, pp. 892-896.
17. M. J. Ashjaee and S. M. Reddy, "On totally self-checking checkers for a separable code," *IEEE Transactions on Computers*, Vol. 26, No. 8, 1977, pp. 737-744.
18. S. J. Piestrak, "Design of fast self-testing checkers for a class of Berger codes," *IEEE Transactions on Computers*, Vol. 36, No. 5, 1987, pp.629-634.

19. T. R. N. Rao, G. L. Feng, M. S. Kolluru and J.-C. Lo, "Novel totally self-checking berger code checker designs based on generalized berger code partitioning," *IEEE Transactions on Computers*, Vol. 42, No. 8, 1993, pp. 1020-1024.
20. T. R. N. Rao, G. L. Feng, M. S. Kolluru and J.-C. Lo, "Correction to novel totally self-checking berger code checker designs based on generalized berger code partitioning," *IEEE Transactions on Computers*, Vol. 43, No. 5, 1994, p.640.
21. S. Kundu, E. S. Sogomonyan and M. Goessel, "Self-checking comparator with one periodic output," *IEEE Transactions on Computers*, Vol. 45, No. 3, 1996, pp. 379-380.

Wen-Feng Chang (張文峰) received the B.S. degree in computer science from Soo Chow University in 1988 and the M.S. degree in information engineering from Feng Chia University in 1991. Currently he is a Ph.D. candidate in the Department of Computer Science, National Tsing Hua University. His research interests include VLSI testing and fault tolerance.

Cheng-Wen Wu (吳誠文) received the BSEE degree in 1981 from National Taiwan University, Taipei, Taiwan, and the MS and Ph.D. degrees, both in electrical and computer engineering, in 1985 and 1987, respectively, from the University of California, Santa Barbara. From 1981 to 1983, he was an ensign instructor at the Chinese Naval Petty Officers' School of Communications and Electronics, Tsoying, Taiwan. From 1983 to 1984, he was with the Information Processing Center of the Bureau of Environmental Protection, Executive Yuan, Taipei, Taiwan. From 1985 to 1987 he was a post graduate researcher at the Center for Computational Sciences and Engineering at UCSB. Since 1988, he has been with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan, where he is currently a professor. He also served as the director of the Computer and Communications Center of NTHU from February, 1996 to February, 1998. Dr. Wu was the Technical Program Chair of the Fifth IEEE Asian Test Symposium (ATS'96). He received the Teaching Award from NTHU in 1996 and the Outstanding Electrical Engineering Professor Award from the Chinese Institute of Electrical Engineers (CIEE) in 1997. He is interested in the relationship between architectures and algorithms with respect to the design and testing of high performance VLSI circuits and systems. He is a member of CIEE and a senior member of IEEE.