1998

A Case Study of Self-Checking Circuits Reliability

Jien-Chung Lo
University of Rhode Island, jcl@ele.uri.edu

Citation/Publisher Attribution
Lo, J.-C. (1998). A Case Study of Self-Checking Circuits Reliability. *VLSI Design*, *5*(4):373-383. doi: 10.1155/1998/71348
Available at: http://dx.doi.org/10.1155/1998/71348

A Case Study of Self-Checking Circuits Reliability

VLSI DESIGN
1998, Vol. 5, No. 4, pp. 373–383
Reprints available directly from the publisher
Photocopying permitted by license only

A Case Study of Self-Checking Circuits Reliability

JIEN-CHUNG LO

Department of Electrical and Computer Engineering, The University of Rhode Island, Kingston, RI 02881-0805

In this paper, we analyze the reliability of self-checking circuits. A case study is presented in which a fault-tolerant system with duplicated self-checking modules is compared to the TMR version. It is shown that a duplicated self-checking system has a much higher reliability than that of the TMR counterpart. More importantly, the reliability of the self-checking system does not drop as sharply as that of the TMR version. We also demonstrate the trade-offs between hardware complexity and error handling capability of self-checking circuits. Alternative self-checking designs where some hardware redundancies are removed with the lost of fault-secure and/or self-testing properties are also studied.

Keywords: Alternative self-checking designs, Duplicated self-checking systems, Reliability analysis, Self-checking circuits, Triple modulo redundancy, Totally self-checking goal

1 INTRODUCTION

The need for self-checking circuits increases as the development of VLSI pushes for lower circuit dimension and higher packaging density. It is well known that these features will make VLSI chips even more vulnerable to transient and soft errors. Concurrent error detecting mechanism is required for a reliable operation.

The self-checking circuits have been widely studied[1–3]. They are designed to perform concurrent error detection in safety critical applications. However, the designers of self-checking circuits and/or systems are more concentrate on the satisfaction of a given set of definitions than the desired application environment and the nature of the mission. In other words, all existing self-checking circuits or checkers have been designed to guarantee the extreme case error handling capability.

There is no one single rule for fault-tolerant computing system design. The main reason is that a fault-tolerant system must be built to fit the targeted applications. For example, FTMP[4] and SIFT[5] are designed for avionics systems. The requirement is to achieve an ultra-reliability, $1-10^{-9}$ within a short flight time (about 10 hours on average). The electronic switching system (ESS)[6], on the other hand, is designed to achieve a high availability. It is no surprise that the design techniques of the above mentioned three systems differ significantly.

Obviously, many possible self-checking circuit designs have yet to be explored. Take totally self-checking (TSC) circuits[2] for example, a TSC circuit must guarantee that no error propagation before a fault detection and that all modeled faults can be detected eventually. The TSC definitions set a very strict standard for high degree of error handling capability. This supreme error handling capability

has been termed TSC goal[3]. However, this level of error protection comes with a cost of high redundancy. One may easily see that the redundancy level of a TSC design can be reduced at a cost of losing the level of error protection. In[7,8], we proposed to calculate the probability that a circuit achieves the TSC goal. This probability can then be used to determine the worthiness of a modification that attempts to reduce the redundancy.

When we use a measurement rather than a set of restricted definitions to evaluate self-checking designs, we may freely explore a wide spectrum of design possibilities. Moreover, a self-checking design that is optimally (or near optimally) tuned for the given application environment can be obtained. In this paper, we present a case study of reliability analysis of self-checking circuits. We will present the evaluation of a self-checking system consists of dual self-checking modules. The duplication of self-checking module provides the concurrent error correction capability and can ensure the continuous operations. This is similar to the stepwise negotiating voting[9]. We compare the reliability of this design to the triple modulo redundancy (TMR) version that can provide the same capability. Moreover, we demonstrate the impact of a design alternative that trades some error handling capability for a lower hardware redundancy level.

2 BACKGROUND

2.1 Definitions

In this paper, we define *defects* as the abnormality in physical layer, *e.g.* bridges and breaks. The *faults* are the abstract view of classifiable defects, *e.g.* stuck-at, stuck-open and stuck-on faults. The *errors* are the erroneous pattern shown at the output of the circuit in the presence of physical abnormality. A *failure* is defined here as the situation where errors are propagated beyond the boundary of the circuit. This is in accordance with the conventional TSC definitions.

TSC circuits and checkers have been formally defined as follows[2]:

D1: A circuit is *fault-secure* with respect to fault set **F**, if and only if for every single fault and for all code word input the circuit will never produce an incorrect code word.

D2: A circuit is *self-testing* with respect to fault set **F**, if and only if each fault in **F** can be detected by at least one code word input.

D3: A circuit is *totally self-checking* if it satisfies D1 and D2.

The TSC goal is stated as follows[3]: "Given the fault assumption, a self-checking circuit always produces a noncode word as the first erroneous output due to a fault." The TSC circuits achieve TSC goal given the following two assumptions[3]:

A1: Each failure can be modeled as a member of the predefined fault set.

A2: Faults occur one at a time, and the time interval between the occurrences of any two faults is sufficiently long so that all input code words are applied to the circuit input.

To satisfy A1, the design must have a 100% fault coverage with respect to the predefined fault set. However, this predefined fault set may not necessarily include all the possible faults. As for A2, we can see that a 100% guarantee of TSC goal is virtually impossible. The time interval between the occurrences of any two faults is a random variable governed by the component failure rate. Since it is a random variable, there is no guarantee that A2 can always be satisfied. The problem is even worse for embedded TSC circuits when some crucial input vectors are not available. These drawbacks are well-known for the TSC circuits.

Strictly speaking, a TSC circuit is no longer TSC when it uses an error control code that is incapable of correcting or detecting *all* possible error types. Also, a TSC circuit is no longer TSC when it is placed in an embedded environment where the possible code word combinations are insufficient to detect all modeled faults. In such cases, we simply call the circuits self-checking (SC).

2.2 Previous Works

In[10], the reliability of a duplicated self-checking system is compared to that of a triple modulo redundancy (TMR) system. This analysis assumes a perfect coverage, *i.e.* no violation of the TSC assumptions may occur. In this paper, we study the reliability comparison using similar configurations. The difference is that, in this paper, the coverage will be included in the reliability evaluations. The coverage is estimated based on the error handling capability of the self-checking circuit.

In most self-checking studies, the size of the minimum test set has been used to compare the self-checking designs. This can be justified from the close examination of the TSC assumptions. However, we need something more concrete to ascertain the dynamic fault and/or error handling performance of the TSC circuits. Only a handful of related works regarding the performance of TSC circuits have been reported in the literature[11–14,7]. The study by Courtois[11] formulates the performance of partially self-checking systems. The TSC circuits are assumed to be governed by a simple failure mechanism.

The studies of Lu and McCluskey[12] and Fujiwara *et al.*[13,14] are aimed at the static behavior rather than run-time behavior of self-checking circuits. Ref. [12] presents two quantitative measures: TIF (test input fraction) and SIF (secure input fraction) to describe the figure of merit for TSC circuits. Ref. [13,14] suggested a measure called *checker fault coverage* (CFC) to describe the fault detection capability of a TSC circuit. These measurements are inadequate to describe the run-time behavior of a self-checking circuit.

Recently, Lo and Fujiwara[7] present a probabilistic measurement for TSC circuits. This measurement provides a steady-state probability, *i.e.* this measurement is not a function of time, of the run-time behavior. In[8], the formula to derive the similar measurement as a function of time is presented. Although some of the derivations in[8] are useful for the reliability analysis, the emphasis of[8] is mainly in the probability to achieve the TSC goal itself. There is no further elaboration on the application of such probability to the reliability analysis of self-checking circuits.

3 RELIABILITY MODELING OF SELF-CHECKING CIRCUITS

Figure 1 shows self-checking system model studied in this paper. Since a self-checking system provides only the concurrent error detecting capability, we need two self-checking system for concurrent error correction. The correct output is selected based on the error indication signals from the two self-checking checkers.

3.1 Notations

The TSC goal[3] has been proposed as a measure of all self-checking circuits given the two fault assumptions[3]. Conventionally, a self-checking circuit must achieve the TSC goal. However, in this paper, we shall use a more relaxed definition and name any circuit that is designed with inherent concurrent error detection capability a self-checking circuit.

The followings are the notations to be used in the derivations.

N: Total number of faults in the circuit. $F = \{f_1, \ldots, f_N\}$.
M: The circuit is fault-secure with respect to FS $= \{f_1, \ldots, f_M\}$, $0 \le M \le N$.
λ_i: The failure rate per cycle of f_i, $1 \le i \le N$.
T_i: Detecting rate of f_i per cycle.
Q_i: $= 1 - T_i$.

For simplexity, we use the single stuck-at fault model. If a more realistic fault model is to be considered, we need to address the fact that some faults such as stuck-open faults need two tests in sequence. We also assume a synchronous circuit operation. The cycle time will be used as the unit of calculations.

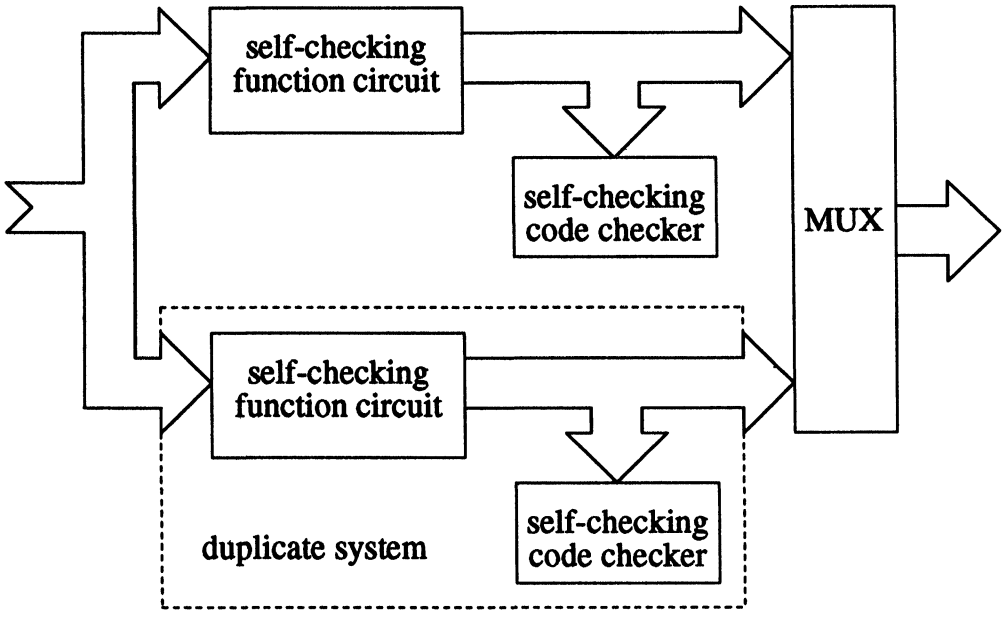

FIGURE 1 The architecture of the reliability case study.

3.2 Reliability Derivation

Conventionally, reliability of a self-checking should be defined as the probability that no fault occur before or on the t^{th} cycle, and thus

$$R(t) = e^{-N\lambda t}. \tag{1}$$

Using the terminology of[15][2], we shall call this the real reliability. The apparent reliability of a self-checking circuit is defined as the probability that the self-checking circuit is still operational at the t^{th} cycle. When a single fault occurs in a self-checking circuits, the circuit will remain operational until the fault detection occurs if the circuit is fault-secure with respect to that fault.

Figure 2 depicts the possible scenarios of a self-checking circuit at the t^{th} cycle. We define two functions: $SC(t)$ and $UN(t)$ to represent the probability of the two cases where a fault occurs and the TSC goal is secured. The derivations of these two functions will be addressed later.

The TSC goal is said lost when either (1) a second fault occurs before the first fault is detected; or (2) the circuit is not fault-secure with respect to the first fault. In the former case, the TSC goal is lost as soon as the second fault occurs. For the later case, the TSC goal is lost as soon as the first fault occurs.

In a duplicated system, such as the one shown in Figure 1, we also need to calculate the probability of a successful error correction. The system is considered reliable if the error is correctly identified and the correction is also correctly performed. Conventionally, the reliability of a duplicated system may be formulated as

$$R_d(t) = R^2(t) + 2CR(t)(1 - R(t)). \tag{2}$$

The coverage[16], C, is defined in this case the probability of a successful error correction. Note that C may be a function of time also. For the system shown in Figure 1, we may see that $UN(t)$ represents a counterpart of the term $C(1-R(t))$ in a

[2] See example 3.25, pp. 164 of[15].

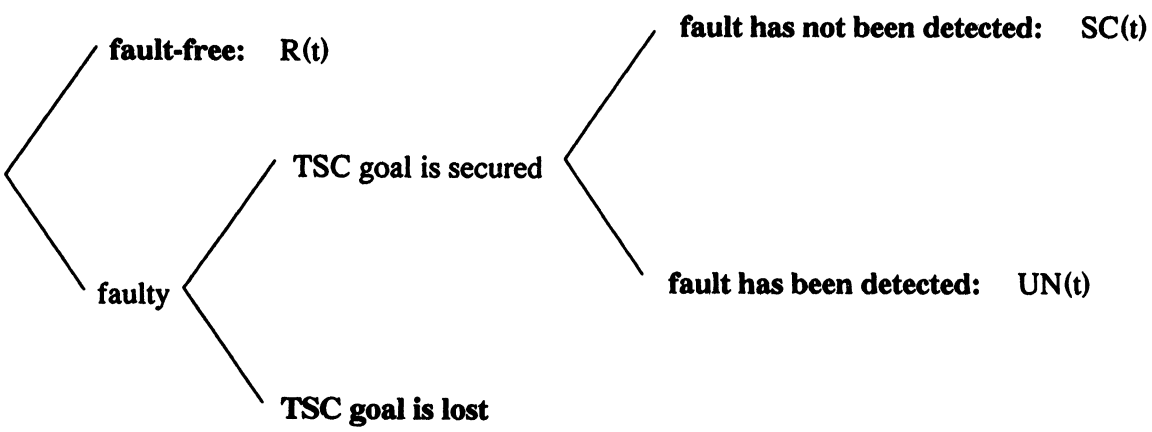

FIGURE 2 The four possible states of a self-checking circuit at the t^{th} cycle.

conventional duplicated system. Therefore, the reliability of the duplicated self-checking system is

$$R_{dup}(t) = R_a^2(t) + 2UN(t)R_a(t).$$

However, since a self-checking functional circuit must always be accompanied by a self-checking checker, we need to adjust the above formula. First the apparent reliability of the combinational of a self-checking circuit and a self-checking checker is

$$R_{a-sc}(t) = R_{a-f}(t)R_{a-c}(t), \tag{3}$$

where R_{a-f} is the apparent reliability of the self-checking functional circuit and R_{a-c} is that of the self-checking checker. As for the probability of a successful error correction given that a single fault occurs for the combination of a self-checking functional circuit and a self-checking checker, we find

$$UN_{sc}(t) = UN_f(t)R_{a-c}(t) + UN_c(t)R_{a-f}(t) + UN_f(t)UN_c(t). \tag{4}$$

Therefore, the reliability of the duplicated self-checking system shown in Figure 1 is

$$R_{dup-sc}(t) = R_{a-sc}^2(t) + 2UN_{sc}(t)R_{a-sc}(t).$$

For a even more detailed analysis, the reliability of the multiplexer in Figure 1 is also taken into account. Therefore,

$$R_{dup-sc}(t) = R_{mux}(t)(R_{a-sc}^2(t) + 2UN_{sc}(t)R_{a-sc}(t)) \tag{5}$$

gives a more accurate reliability measurement.

3.3 Derivation of $R_a(t)$

The following equations have also been presented in[8]. However, since the measurement of interest in[8] is the probability to achieve the TSC goal, the formula is provided for $S(t) = SC(t) + UN(t)$. The $SC(t)$ and $UN(t)$ have never been separately derived.

The apparent reliability of a self-checking circuit can be computed as follows.

$$\begin{aligned} R_a(t) &= R(t) + SC(t) \\ &= e^{-N\lambda t} + \sum_{i=1}^{M}\sum_{j=1}^{t} \lambda e^{-N\lambda t} Q_i^{t-j+1} \\ &= e^{-N\lambda t} + \sum_{i=1}^{M} \lambda e^{-N\lambda t} \frac{(Q_i - Q_i^{t+1})}{T_i}. \end{aligned} \tag{6}$$

We assume that all faults have equal arrival rate and are governed by exponential distribution law.

We note that,

$$\begin{aligned} SC(t) &= P(\text{achieve TSC goal without a detection}) \\ &= \sum_{j=1}^{t} P(\text{the first fault occurs at the } j^{th} \text{ cycle}) \\ &\quad \times P(\text{no other fault occurs between the } j^{th} \text{ and the } t^{th} \text{ cycles}) \\ &\quad \times P(\text{the first fault is not yet detected at the } t^{th} \text{ cycle}) \end{aligned} \tag{7}$$

In the above equation, the assumption is that the TSC goal is lost whenever a second fault occurs. This is not suitable for circuits designed as strongly fault-secure (SFS)[3] or strongly code disjoint (SCD)[17]. The reliability analysis of SFS and SCD designs will require the explicit study of the fault sequences instead of faults.

Since

$$\begin{aligned} &P(\text{the first fault occurs at the } j^{th} \text{ cycle}) \\ &\quad = P(\text{a fault occurs at the } j^{th} \text{ cycle}) \\ &\qquad \times P(\text{this fault does not occur in any other cycle before the } j^{th} \text{ cycle}) \\ &\qquad \times P(\text{the rest of the faults do not occur before or on the } j^{th} \text{ cycle}) \\ &\quad = \lambda e^{-\lambda} \times e^{-\lambda(t-1)} \times e^{-(N-1)\lambda j} \\ &\quad = \lambda e^{-\lambda(t-j)} \times e^{-N\lambda j}, \end{aligned} \tag{8}$$

$$P(\text{no other fault occurs between the } j^{th} \text{ and the } t^{th} \text{ cycles}) = e^{-(N-1)\lambda(t-j)}, \tag{9}$$

and

$$P(\text{the first fault is not yet detected at the } t^{th} \text{ cycle}) = Q_i^{t-j+1}, \tag{10}$$

we may easily verify Equation (6).

The non-fault-secure faults are excluded from the derivation of $SC(t)$. This implies that we consider the TSCG goal is lost whenever a non-fault-secure fault appears. If the circuit is not self-testing with respect to a particular fault, say f_i, that fault has a $T_i = 0$. We may easily see that this fault will not contribute to the $SC(t)$ at all. In other words, we also assume that the TSC goal is lost whenever a non-self-testing fault occurs. We should remark here that this is a worst case assumption.

3.4 Derivation of $UN(t)$

The $UN(t)$ of a self-checking circuit can be formulated as follows.

$$\begin{aligned} UN(t) = \sum_{i=1}^{M} \sum_{j=1}^{t} & P(f_i \text{ occurs at the } j^{th} \text{ cycle}) \\ & \times P(\text{no other fault occurs before or on the } j^{th} \text{ cycle}) \\ & \times \Big(\sum_{k=j}^{t} P(f_i \text{ is detected at the } k^{th} \text{ cycle}) \\ & \times P(\text{no other fault occurs between the } j^{th} \text{ cycle and the } k^{th} \text{ cycle}) \Big). \end{aligned} \tag{11}$$

We assume here that at each cycle each input has the same probability of occurrence. Therefore, the probability that f_i occurs at the j^{th} cycle and is detected at the k^{th} cycle, $k \le j$, is $(1-T_i)^{k-j}T_i$. T_i is the probability that f_i can be detected at each cycle. In other words, T_i is the sum of arrival rates of all code word inputs that test f_i. Since the fault detection occurs at the k^{th} cycle, there are $k-j$ cycles the circuit receives inputs that cannot detect f_i. In other words, the detection of a fault is a geometric distribution.

$$P(f_i \text{ is detected at the } k^{th} \text{ cycle}) = Q_i^{k-j} T_i. \tag{12}$$

The discussions given above and in the previous section shows that

$$\begin{aligned} UN(t) &= \sum_{i=1}^{M} \sum_{j=1}^{t} \lambda e^{-\lambda(t-j)} e^{-N\lambda j} \left(\sum_{k=j}^{t} \frac{Q_i^{k-j} T_i e^{-N\lambda(k-j)}}{e^{-\lambda(k-j)}} \right) \\ &= \sum_{i=1}^{M} \lambda \Big\{ \Big[e^{-\lambda(t+1)} e^{-N\lambda} T_i - e^{-\lambda} e^{-N\lambda(t+1)} \\ &\quad \times (1 - Q_i^{t+1}) + e^{-N\lambda(t+2)} (Q_i - Q_i^{t+1}) \Big] \Big/ \\ &\quad \Big[(e^{-\lambda} - e^{-N\lambda} Q_i)(e^{-\lambda} - e^{-N\lambda}) \Big] \Big\}. \end{aligned} \tag{13}$$

4 A CALCULATION EXAMPLE

In this section, we shall use the formulas presented earlier to evaluate self-checking circuit designs. We shall use the well-known parity encoded adder[18,19] as an example.

4.1 A TSC Parity Encoded Adder

The parity encoded adders in[18] require duplicated carry units to generate the redundant carry bits for the calculation of the output parity bit. This is because some faults in the adder may induce multiple unidirectional errors on the carry bits, which result in arithmetic errors on the output bits. These errors are undetectable by the parity code unless the duplicated carry bits are provided. Figure 3 shows a 4-bit parity encoded adder as in[18,19]. We assume a single stuck-at fault model such that the fault set consists of all possible single stuck-at faults. Table I lists all the possible faults sorted by the number of their test inputs. In this case, N = 202, and M = 202. The T_i's are computed assuming that each input has equal probability of occurrence.

In order to clearly demonstrate the calculation of T_i, we show in Table II the tests for each fault in a full adder. The logic diagram with corresponding node names of a full adder is shown in Figure 4.

TABLE I Numbers of faults grouped by T_i for circuit in Figure 3

T_i	
non-fault-secure	0
64/512	105
128/512	4
192/512	4
256/512	89
Total	202

TABLE II Complete tests for stuck-at faults of full adder in Figure 4

fault	X	Y	C_i	no. of tests	fault	X	Y	C_i	no. of tests
a s-a-0	1	X	X	4	a s-a-1	0	X	X	4
b s-a-0	X	1	X	4	b s-a-1	X	0	X	4
c s-a-0	0	1	X	4	c s-a-1	0	0	X	4
	1	0	X			1	1	X	
d s-a-0	X	X	1	4	d s-a-1	X	X	0	4
e s-a-0	0	0	1	4	e s-a-1	0	0	0	4
	0	1	0			0	1	1	
	1	0	0			1	0	1	
	1	1	1			1	1	0	
f s-a-0	1	0	1	1	f s-a-1	0	0	1	1
g s-a-0	1	0	1	1	g s-a-1	1	0	0	1
h s-a-0	1	1	0	1	h s-a-1	0	1	0	1
i s-a-0	1	1	0	1	i s-a-1	1	0	0	1
j s-a-0	0	1	1	1	j s-a-1	0	0	1	1
k s-a-0	0	1	1	1	k s-a-1	0	1	0	1
l s-a-0	1	0	1	1	l s-a-1	0	X	0	4
						0	0	1	
						1	0	0	
m s-a-0	1	1	0	1	m s-a-1	0	0	X	4
						0	1	0	
						1	0	0	
n s-a-0	0	1	1	1	n s-a-1	X	0	0	4
						0	0	1	
						0	1	0	
o s-a-0	0	1	1	4	o s-a-1	0	0	X	4
	1	0	1			0	1	0	
	1	1	X			1	0	0	

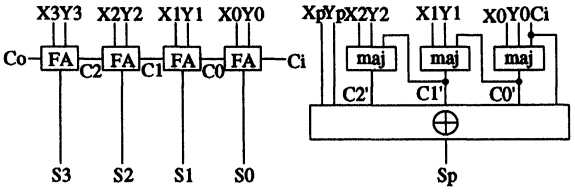

FIGURE 3 The classical implementation of TSC parity encoded adder with duplicated carry unit.

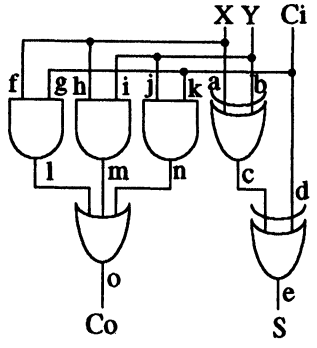

FIGURE 4 A gate level logic diagram of a full adder with marked stuck-at faults.

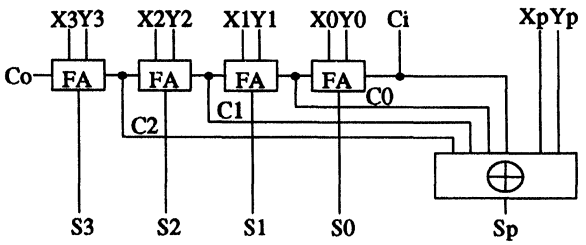

FIGURE 5 An alternative self-checking parity encoded adder implementation without the duplicated carry unit.

TABLE III Numbers of faults grouped by T_i for circuit in Figure 5

T_i	
non-fault-secure	80
64/512	0
128/512	4
192/512	4
256/512	54
Total	142

4.2 A Self-Checking Parity Encoded Adder

The parity encoded adder in Figure 3 uses a duplicated carry unit. This design is necessary to avoid miss detection due to error propagation on the carry chain[19]. A possible design alternative here is to reduce the complexity by removing the duplicated carry unit. The internal carry bits are used instead to generate the output (sum) parity bit. This design is no longer TSC and thus we will refer to it as self-checking only. This self-checking adder is shown in Figure 5.

Table III summaries all the possible stuck-at faults in Figure 5. For this circuit, we find $N = 142$ and $M = 62$, because there are 80 non-fault-secure faults. Obviously that this alternate design has poorer error handling capability. However, we also know that this alternate design has a reduced hardware complexity or a smaller N. It should be noted here that there is no direct relationship between the degree of lost error handling capability and the amount of hardware redundancy reduced.

4.3 Comparison

In this section, we compare the reliability of the structure shown in Figure 1 and its TMR counterpart. The TSC parity adder and the self-checking adder discussed previously will be used here. The reliability of a TMR adder is

$$R_{\text{TMR}} = R_v(3R^2 - 2R^3) \tag{14}$$

where R_v is the reliability of the voter.

Figure 6 shows the reliability plots of the duplicated self-checking system with TSC parity adder and the self-checking parity adder, respectively, and that of a TMR adder. Note that the parity checker associated with the parity encoded adder is also taken into account.

First, we observe that the duplicated system with TSC parity adder gives the highest reliability. Of course, in this example, the TMR version uses the highest level of hardware redundancy. The TSC adder version uses 178 additional hardware unit

(per fault), the self-checking adder uses 58 additional hardware unit, and the TMR version uses an extra of 320 units. The above numbers include the checkers, voters, and the multiplexers. We may expect that the TSC duplicated system with TSC components to always perform better than the TMR version when the hardware redundancy level is comparable. This can be observe from Figure 6 that the curve of the TMR version drops much more rapidly.

As for the alternate self-checking design, we know that the inherent reliability of the circuit is increased due to the reduction of hardware complexity. This is at a cost of losing a significant level of error handling capability. We see from Table III that 80 out of a total of 142 faults, or about 56%, are non-fault-secure.

From Figure 6, we see the effect of this significant lost of error handling capability. The reliability of the self-checking alternative is clearly much lower that that of the TSC. Nonetheless, we also see that this self-checking alternative is not totally useless. When $N\lambda t < 0.07$, the reliability of the self-checking version is only slightly lower than that of the TMR version. When $N\lambda t > 0.07$, the self-checking version has a higher reliability than the TMR version. More importantly, the gap between the two reliabilities increases as $N\lambda t$ increases.

A fault-tolerant system is application oriented. A good fault-tolerant design is a design that is optimized for its application. If the application requires that the system to have a reliability greater than 0.9 for $N\lambda t \leq 0.1$, then obviously the self-checking version is the best choice.

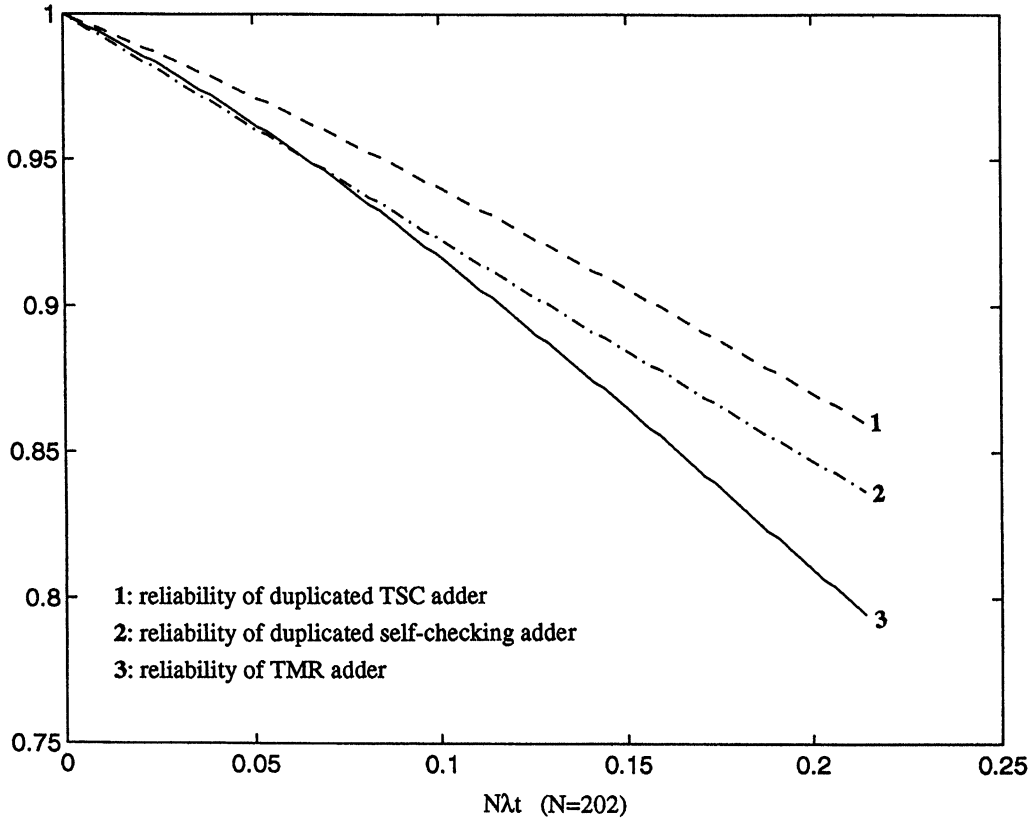

FIGURE 6 The reliability of a duplicated system with TSC parity encoded adders, a duplicated system with self-checking parity encoded adders, and a TMR system with triplicated adders and voter.

5 CONCLUSIONS

We have presented in this paper the reliability analysis of self-checking systems. We also show that alternate design method can be justified based on the reliability modeling rather than the theoretical soundness. The evaluations shown in this paper give the worst case numbers. The reason is that we assume the fault-secure property is lost whenever: (1) a non-fault-secure fault occurs or (2) a second fault occurs before the first fault is detected. Obviously, the reliability derived based on these assumptions is lower than the realistic reliability. Even if a non-fault-secure fault occurs as the first fault, the fault-secure property is lost only when the appropriate code word input is also presented. Further, a self-checking circuit may exhibit SFS and/or SCD properties for some fault sequences. A circuit is SFS or SCD if and only if all fault sequence is SFS or SCD. In this case, the presented derivation under estimates the probability that the fault-secure property is still intact.

For a more accurate estimation of self-checking reliability, we must handle these two cases explicitly. The first case can be easily incorporated by counting the probability of losing fault-secure property for each non-fault-secure fault and use it in the equation. This probability can be derived as a ratio of the number of code word inputs that the circuit will not lost the fault-secure property. To cope with the second case, one must analyze in detail all possible fault sequences for their fault-secureness implications, as we have pointed out in Section 3.3. These will be the subjects of future studies.

Finally, we point out an important implication of this work is that a self-checking design that does not guarantee to be fault-secure and/or self-testing can still be used in some applications. After all, the optimal fault-tolerant design is the one that has been optimized for the given application. This is an important consideration that should be included in the self-checking circuits research.

Acknowledgments

This work is supported by the National Science Foundation under grant MIP-9308085.

References

[1] W. C. Carter and P. R. Schneider, "Design of dynamically checked computers," in *Proc. IFIP-68*, pp. 878–883, August 1968.

[2] 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, March 1973.

[3] J. E. Smith and G. Metze, "Strongly fault secure logic networks," *IEEE Trans. Comput.*, vol. C-27, pp. 491–499, June 1978.

[4] A. L. Hopkins, Jr., T. B. Smith III and J. H. Lala, "FTMP—A highly reliable fault-tolerant multiprocessor for aircraft," *Proc. IEEE*, vol. 66, pp. 1221–1239, October 1978.

[5] J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliarsmith, R. E. Shostak and C. B. Weinstock, "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," *Proc. IEEE*, vol. 66, pp. 1240–1255, October 1978.

[6] W. N. Toy, "Fault-tolerant design of local ESS processors," *Proc. IEEE*, vol. 66, pp. 1126–1145, October 1978.

[7] J. C. Lo and E. Fujiwara, "A probabilistic measurement for totally self-checking circuits," in *Proc. IEEE Workshop on Defect and Fault Tolerance in VLSI*, pp. 263–270, October 1993.

[8] J. C. Lo and E. Fujiwara, "Probability to achieve TSC goal," *submitted to IEEE Trans. Comput.*

[9] T. Takano, T. Yamada, K. Shutoh and N. Kanekawa, "Fault-tolerant experiments of the "Hiten" onboard space computer," in *Proc. 21st Int'l Symp. Fault-Tolerant Comput.*, pp. 26–33, June 1991.

[10] M. Lubaszewski and B. Courtois, "Reliable fail-safe systems," in *Proc. 2nd Asian Test Symp.*, pp. 32–37, November 1993.

[11] B. Courtois, "Performance modeling of partially self-checking systems," in *Proc. 12th Symp. Fault-Tolerant Comput.*, pp. 140–146, June 1982.

[12] D. J. Lu and E. J. McCluskey, "Quantitative evaluation of self-checking circuits," *IEEE Trans. Computer-Aided Designs*, vol. CAD-3, pp. 150–155, April 1984.

[13] E. Fujiwara, N. Mutoh and K. Matsuoka, "A self-testing group-parity prediction checker and its use for built-in testing," *IEEE Trans. Comput.*, vol. C-33, pp. 578–583, June 1984.

[14] E. Fujiwara and K. Matsuoka, "A self-checking generalized prediction checker and its use for built-in testing," *IEEE Trans. Comput.*, vol. C-36, pp. 86–93, January 1987.

[15] K. S. Trivedi, *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice-Hall, Inc., Englewood Cliffs, NJ (1982).

[16] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems: Design and Evaluation*. 2nd edition, Digital Press (1992).

[17] M. Nicolaidis and B. Courtois, "Strongly code disjoint checkers," *IEEE Trans. Comput.*, vol. 37, pp. 751–756, June 1988.

[18] F. F. Sellers, M. Y. Hsiao and L. W. Bearnson, *Error Detecting Logic for Digital Computers*. McGraw-Hill, New York (1968).
[19] T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*. Prentice Hall, Englewood Cliffs, New Jersey (1989).

Author's Biography

Jien-Chung Lo received his M.S. and Ph.D., both in Computer Engineering, from University of Southwestern Louisiana in 1987 and 1989, respectively. He is currently an Associate Professor at the University of Rhode Island. He served on the Program Committees of *1994 IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems* and the *1st IEEE International On-Line Testing Workshop*. His research interests include: designs and evaluations of self-checking circuits and systems, dependable distributed computing systems, and VLSI defect modeling and testing.