EFFICIENT REALIZATION OF A THRESHOLD VOTER FOR SELF-PURGING REDUNDANCY[1]

José M. Quintana, María J. Avedillo, and José L. Huertas

Instituto de Microelectrónica de Sevilla, IMSE-CNM, Univ. de Sevilla.

Edif. CICA, Avda. Reina Mercedes s/n, Sevilla 41012, SPAIN

[1]This effort was partially supported by the spanish CICYT under Project TIC97-0648.

Indexing Terms: Self-purging Redundancy, Fault Tolerance, Sorting Networks, Threshold Logic.

***Abstract**:*

The self-purging technique is not commonly used mainly due to the lack of practical implementations of its key component, the threshold voter. A very efficient implementation of this voter is presented which uses a decomposition technique to substantially reduce the circuit complexity and delay, as compared to alternative implementations.

Introduction

Achieving ultrareliable systems is a goal of increasing importance in many areas. The self-purging hybrid approach [1] allows us to improve the reliability of digital systems and it has very simple switching mechanisms, a straighforward design and is simpler, cheaper and more reliable than other hybrid approaches. Also, the faulty modules can be replaced from the system without affecting the normal operation of the system. This characteristic of self-purging systems is extremely attractive because it eases maintenance and repair.

In the self-purging approach each of the *N* identical modules has the capability to disconnect itself from the system in the event that its output disagrees with the voted output of the system. Switches remove their associated module from the system in case the module fails. The voter provides the mean for the masking of any fault that occurs in the module outputs. The voting is done by means of a threshold gate. The choice of the voter threshold is critical for the tolerance of self-purging systems to multiple faults. In general, the best voter threshold is equal to half the number of the remaining fault-free modules [1].

In spite of the advantages described, the self purging technique is not commonly used in fault-tolerant digital designs. One reason for this limited utilization is that it is difficult to construct a threshold gate because it is claimed as an analog element, which is not practical for digital use. Concerning digital implementations, a direct implementation of the voter can be very expensive in terms of the number of gates [2]. Therefore, some simplifying solutions have been reported. Razavi [3] substitutes the threshold gate with a digitally constructed "strong majority voter" which automatically adjusts its threshold. However, the system presented is of limited use because only single module failures per clock cycle are allowed. A solution which can tolerate multiple module failures at any time is presented in [4], but no solution to build the voter is presented. In this paper a very efficient decomposition technique for the logic function performed by the voter is developed, allowing an extremely compact realization.

The Losq Voter

The voter that produces the system output and provides fault masking is a threshold gate. A threshold gate (TG) has n two-valued inputs $x_1, x_2, \ldots, x_n$ and a single two-valued output, y. It is defined by n+1 real numbers: threshold T and weights $w_1, w_2, \ldots, w_n$, being denoted as $[w_1, w_2, \ldots, w_n; T]$, where weight w_i is associated with variable x_i. The input-output relation of a TG is defined as $y=1$ *iff* $\sum_{i=1}^{n} w_i x_i \geq T$ and $y=0$ otherwise. Sum and product in the previous definition are the conventional operations, instead of logical ones.

The classical solution to the automatic threshold adjustment uses a simple mechanism as shown in Figure 1 [1]. The threshold voter is described as $[1, 2, 1, 2, \ldots, 1, 2; N]$, and it has been represented by a non-standard but widely used symbol.

Implementing the Losq voter is a difficult task because the large number of inputs it exhibits. As weights for x_i are 1 and weights for y_i are 2, the total weight for the gate is $3N$. The most economical solutions for implementing this kind of voters are based on sorting networks [2]. Thus, the Losq voter would require a $3N$-input SN to be implemented.

Proposed Voter Implementation

In this section, an efficient and extremely compact decomposition technique for the logic function performed by the voter is developed.

Theorem 1: The output of the voter from a self-purging scheme with N replicated modules can be obtained from a two-level network composed by two N-input SNs, and a combinational network L, with $2\left\lceil \frac{N}{2} \right\rceil$ inputs, as shown in Figure 2a. The first SN depends on y_i variables (i=1, ..., N), and the second one depends on x_i variables (i=1, ..., N).

Proof: Let us consider the Losq voter for N modules given by $[1, 2, \ldots, 1, 2; N]$, *i.e.* the voter output is asserted if $\sum_{i=1}^{N} (x_i + 2y_i) \geq N$. Let us divide the input set $\{x_1, y_1, \ldots, x_N, y_N\}$ into two disjoint subsets, $S_1 = \{y_1, \ldots, y_N\}$, and $S_2 = \{x_1, \ldots, x_N\}$. Let $a = \left\lceil \frac{N}{2} \right\rceil$ and $b = 2 + \left\lfloor \frac{N}{2} \right\rfloor - \left\lceil \frac{N}{2} \right\rceil$ be two quantities which will be very useful in the following. Input combinations which assert voter output L are:

{(a or more variables in S_1 are at logic 1) or
(a-1 or more variables in S_1 at logic 1) and (b or more variables in S_2 at logic 1) or
(a-2 or more variables in S_1 at logic 1) and (b+2 or more variables in S_2 at logic 1) or
.....
(1 or more variables in S_1 at logic 1) and (N-2 or more variables in S_2 at logic 1) or
(N variables in S_2 at logic 1)}

This formulation yields the logic expression for the voter output:

$$f_L = TY_a^N + TY_{a-1}^N \cdot TX_b^N + TY_{a-2}^N \cdot TX_{b+2}^N + \ldots + TY_1^N \cdot TX_{N-2}^N + TX_N^N \qquad (1)$$

where TY_r^N stands for a function which asserts when at least r variables in S_1 are 1, and TX_r^N has a similar definition but referred to S_2. But this definition is exactly that corresponding to an N-input threshold function (inputs from S_1 or S_2) with all the N weights equal to 1 and threshold in r. There are $2a$ of such TGs, a of them depending on variables x_i and another a depending on variables y_i. The fan-in of all of them is N, as shown in Figure 2b.

The solution given in Figure 2a is obtained when the conceptual link between threshold gates and SNs [5] is applied. An n-input SN is a switching network with n outputs which are a sorted (non-increasing order) permutation of the inputs. The set of outputs in an n-input SN are n TGs corresponding to $(T_1^n, T_2^n, \ldots, T_n^n)$, where $T_m^n = 1$ if $\sum_{i=1}^{n} x_i \geq m$, m= 1, 2, …, n, and 0

otherwise. Thus, the outputs from both sets of TGs in the first level shown if Figure 2b are $\left\lceil \frac{N}{2} \right\rceil$ specific outputs of two SNs: one of the SNs sorts y_i variables (i=1, ..., N) and the other sorts x_i variables (i=1, ..., N). ■

To evaluate the performance of both the traditional and the proposed solutions a comparison of their complexity and delay is in order. Concerning the complexity, the number of 1-bit comparators (a 1-bit comparator is implemented by means of a 2-input AND gate and a 2-input OR gate) needed in a n-input SN based on Batcher's odd-even merge sort is given in [6]:

$$C_{sort}(n) = C_{sort}\left(\left\lceil \frac{n}{2} \right\rceil\right) + C_{sort}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + C_{merge}\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right) \quad ; \quad (n > 2) \tag{2}$$

where $C_{merge}\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right)$ is the number of comparators required to merge two sorted sequences of sizes $\left\lceil \frac{n}{2} \right\rceil$ and $\left\lfloor \frac{n}{2} \right\rfloor$, given by:

$$C_{merge}\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right) = C_{merge}\left(\left\lceil \frac{\left\lceil \frac{n}{2} \right\rceil}{2} \right\rceil, \left\lceil \frac{\left\lfloor \frac{n}{2} \right\rfloor}{2} \right\rceil\right) + C_{merge}\left(\left\lfloor \frac{\left\lceil \frac{n}{2} \right\rceil}{2} \right\rfloor, \left\lfloor \frac{\left\lfloor \frac{n}{2} \right\rfloor}{2} \right\rfloor\right) + \left\lceil \frac{n}{2} \right\rceil - 1 \tag{3}$$

and $C_{merge}(1, 1) = 1$, $C_{merge}(2, 1) = 2$ and $C_{sort}(2) = 1$.

Concerning the number of gate levels, the delay time in Batcher's sorting method for n elements is given by:

$$delay(n) = \binom{1 + \lceil \log_2 n \rceil}{2} \tag{4}$$

When these expressions are applied to a self-purging redundancy scheme for N-modules, the cost of the voter is given by $Cost_1 = C_{sort}(3N)$, and its delay is $Delay_1 = \binom{1 + \lceil \log_2 3N \rceil}{2}$. In our approach, the cost of the voter is given by $Cost_2 = 2 \cdot C_{sort}(N) + Cost_{CC}$, $Cost_{CC}$ is the cost of the combinational circuit at the output. The delay is $Delay_2 = \binom{1 + \lceil \log_2 N \rceil}{2} + delay_{CC}$, where $delay_{CC}$ corresponds to the combinational output circuit. Figure 3a clearly shows the difference in complexity between both solutions for typical values of the module number, N, and Figure 3b compares the delay for both approaches for the same range of N. Delay of the combinational output circuits depends on the available gates. For a comparison, we have supposed that 4-input OR gates are available.

Example 1: Let us consider the design of a threshold voter from a self-purging scheme with 5 replicated modules. The voter needed is $[1, 2, 1, 2, 1, 2, 1, 2, 1, 2; 5]$, *i.e.*, the voter output is asserted when $\sum_{i=1}^{5} (x_i + 2y_i) \geq 5$.

From definitions for a and b in Theorem 1, we obtain $a = \left\lceil \frac{5}{2} \right\rceil = 3$ and $b = 2 + \left\lfloor \frac{5}{2} \right\rfloor - \left\lceil \frac{5}{2} \right\rceil = 1$. A logical expression for the voter output in Eq. (1) gives:

$$f_L = TY_3^5 + TY_2^5 \cdot TX_1^5 + TY_1^5 \cdot TX_3^5 + TX_5^5$$

The sum-of-product for this function has $\binom{5}{3} + \binom{5}{2} \cdot \binom{5}{1} + \binom{5}{1} \cdot \binom{5}{3} + \binom{5}{5} = 111$ product terms, of three, four and five literals. However, when the voter is implemented as a sorting network, Eq. (2) gives a cost of $C_{sort}(3 \times 5) = 59$ 2-input comparators. The implementation obtained following the decomposition proposed in Theorem 1 provides a cost of $2 \times C_{sort}(5) = 18$ 2-input comparators plus the cost of the combinational circuit at the output, which can be implemented by two 2-input AND gates and one 4-input OR gate, as shown in Figure 4. Further reduction in the number of comparators in both SN-based solutions can be obtained by eliminating comparators which are not used, but exact figures do not substantially modify the above reasonings about complexity. Concerning the delay time of both the traditional and the proposed solutions, Eq. (4) gives $delay(15) = \binom{1 + \lceil \log_2 15 \rceil}{2} = 10$ time units for the first implementation, and $delay(5) = \binom{1 + \lceil \log_2 5 \rceil}{2} = 6$ time units (5-input SN) plus two time units (combinational part) for the proposed implementation. ■

Conclusions

A low-cost implementation of the classical self-purging approach from Losq has been proposed. It is based on both the application of a decomposition technique and the use of the conceptual link between TGs and SNs. The proposed implementation exhibits an excellent performance when compared to the original solution.

References

[1] J. Losq: "A highly efficient redundancy scheme: self-purging redundancy", *IEEE Trans. on Computers*, vol. C-25, No. 6, pp. 569-578, June 1976.

[2] B. Parhami: "Voting networks", *IEEE Trans. on Reliability*, vol. 40, No. 3, pp. 380-394, Aug. 1991.

[3] H.M. Razavi: "Self-purging redundancy with automatic threshold adjustment", *IEE Proc.-G*, vol. 140, No. 4, pp. 233-236, August 1993.

[4] C.W. Chiou and T.C. Yang: "Self-purging redundancy with adjustable threshold for tolerating multiple module failures", *Electron. Lett.*, 1995, **30**, (11), pp. 930-931.

[5] E.A. Lamagna, "The complexity of monotone networks for certain bilinear forms, routing problems, sorting, and merging", *IEEE Trans. on Comput.*, vol. C-28, pp. 773-782, Oct. 1979.

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

FIGURES

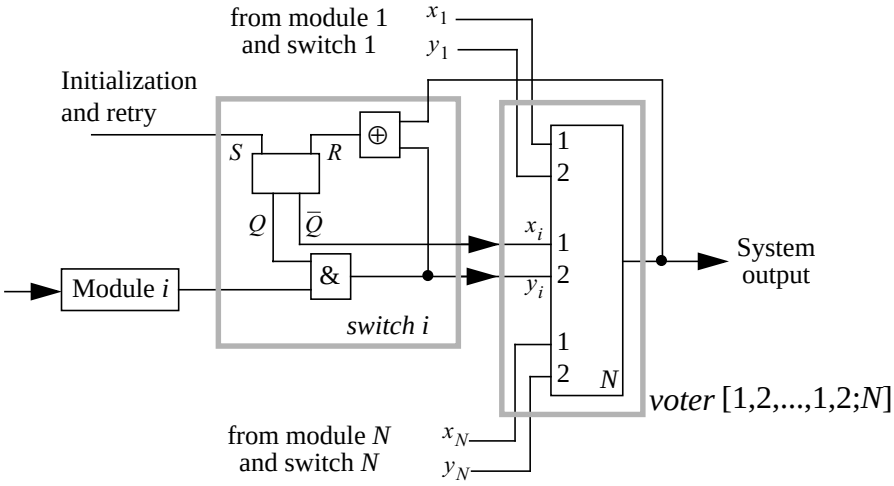
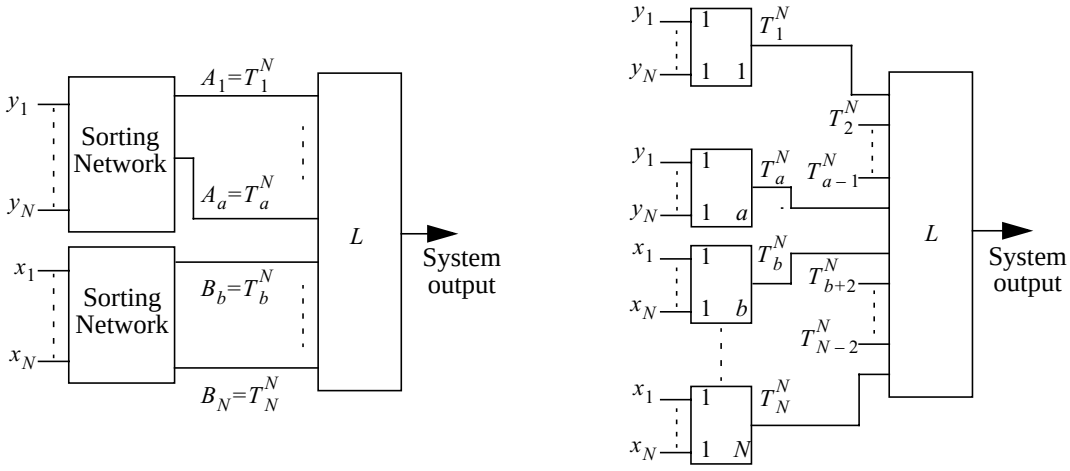
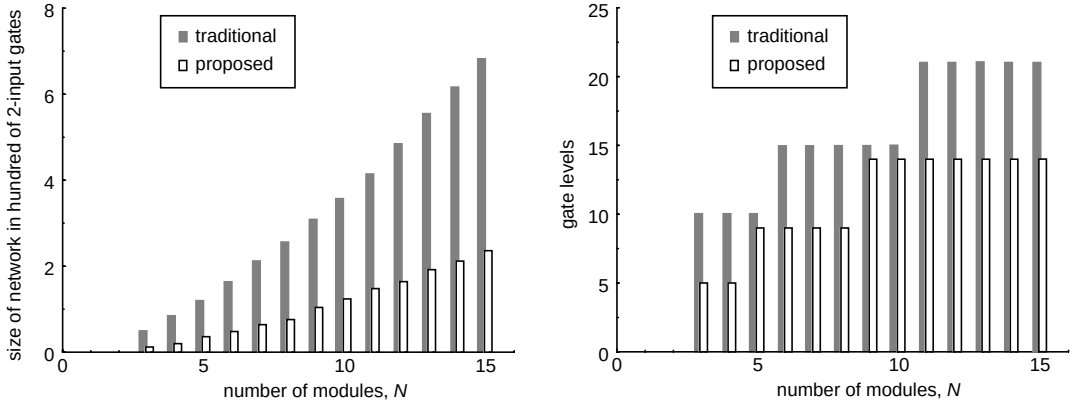


Figure 1:Self-purging approach with *N* modules for optimal tolerance to multiple failures



(a) (b)

Figure 2: Two level realization of self-purging voter, (a) using SNs, (b) using TGs



(a) (b)

Figure 3: Cost parameters for the traditional and the proposed solutions
(a) Size of network in hundred of 2-input gates, (b) Delay

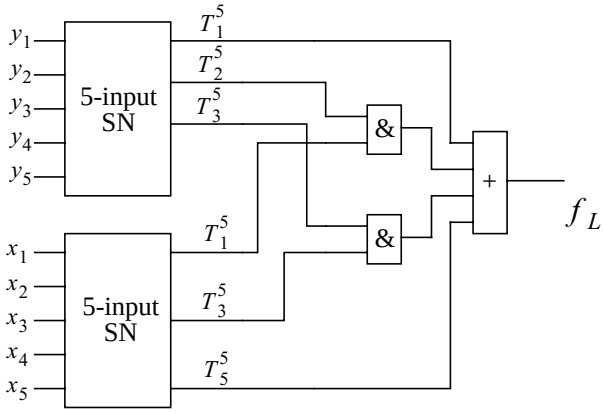


Figure 4: Proposed realization for the self-purging voter in Example 1