Proceedings of the 8th IEEE Mediterranean Conference
on Control and Automation (MED 2000)
Rio Patras, GREECE 17-19 July, 2000

FAULT-TOLERANT SEQUENCE ENUMERATORS

CHRISTOFOROS N. HADJICOSTIS

University of Illinois at Urbana-Champaign, Room 148 Computer and Systems Research Laboratory, 1308 West Main Street, Urbana, IL 61801, USA. Email: chadjic@uiuc.edu.

Abstract. Modular redundancy, the traditional approach to fault tolerance, is prohibitively expensive because of the overhead in replicating the hardware. In this paper we discuss alternative techniques for fault tolerance in sequence enumerators that are implemented as linear finite-state machines (LFSM's). Our approach replaces a given LFSM with a larger, redundant LFSM that preserves the evolution and properties of the original one. The state of the larger LFSM is a linearly encoded version of the state in the original machine and allows an external mechanism to perform error detection and correction by identifying and analyzing violations of the code restrictions. In this paper, we characterize the class of appropriate redundant LFSM's and demonstrate a variety of possibilities for fault tolerance, ranging from no redundancy to full replication.

Key Words: Fault tolerance, linear finite-state machines, concurrent error detection and correction.

1. INTRODUCTION

The traditional, but rather inefficient, way of designing fault-tolerant systems is to use modular redundancy: by replicating the original system we perform the desired function multiple times in parallel; the outputs of all replicas can then be compared (via an external voting mechanism) and the final result can be chosen based on what the majority of the replicas agrees upon. In an effort to utilize redundancy in more efficient ways, a number of researchers have focused on using coding techniques to achieve fault tolerance. Examples of such research work include arithmetic codes [1], and algorithm-based fault tolerance (ABFT) techniques, [2]. More broadly applicable and systematic approaches for introducing redundancy in computations that can be modeled as abelian groups, semigroups or semirings have been studied in [3, 4].

In this paper we discuss fault-tolerance in linear finite-state machines (LFSM's) with particular focus on sequence enumerators. Our approach, which was developed in [5] for general dynamic systems[1], consists of mapping the state of a given LFSM into the higher dimensional state space of a larger, redundant LFSM in a way that preserves the evolution and properties of the original one. By detecting and analyzing violations on the enforced state code, an external mechanism can detect and correct failures that corrupt the state of the redundant machine. Depending on the machine implementation, these violations could be the result of hardware, software or communication failures.

In this paper we assume that the error detection/correction procedure is fault-free[2] and focus on the structure of the corresponding redundant LFSM's. By formulating the problem of constructing redundant implementations as an embedding problem, we solve it in a systematic fashion and completely characterize the class of appropriate redundant LFSM's. This characterization exposes flexibilities that were not considered in previous work, the implications of which are demonstrated in this paper for the case of fault-tolerant sequence enumerators that are built out of unreliable 2-input XOR gates and single-bit memory elements. Fault-tolerant sequence enumerators can be useful in imple-

[1] This technique has also been used to construct fault-tolerant linear time-invariant dynamic systems and Petri nets, [6, 7].

[2] This (common) assumption is reasonable if the complexity of error detection/correction is considerably less than the complexity of the state evolution mechanism of the system, [6]. In [5] we extended this approach to also handle failures in the error-correcting mechanism.

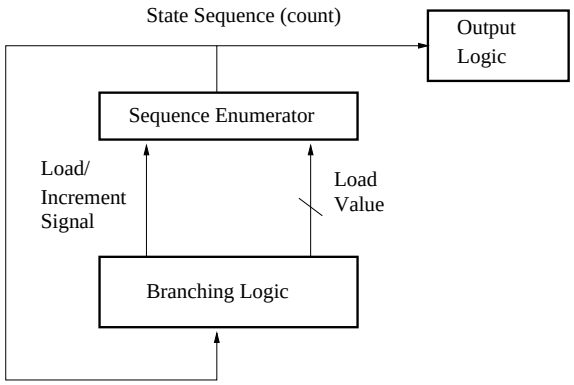

Fig. 1: Digital controller implementation based on a loadable sequence enumerator.

menting reliable digital controllers as shown in Fig. 1 for a loadable sequence enumerator. The techniques that we discuss here are appropriate for handling errors in the "counting" or in the "sequence enumeration". Fault tolerance in digital controllers is a rich topic by itself (see, for example, [8, 9] and references therein) and we will address it explicitly in future research work.

This paper is organized as follows. In Section 2 we provide an introduction to LFSM's and characterize the class of appropriate redundant embeddings for a given machine. In Section 3 we discuss the consequences of our approach for the case of sequence enumerators that are implemented using XOR gates and single-bit memory elements. We conclude in Section 4 with a discussion of future research directions.

2. INTRODUCTION TO LFSM'S AND SEQUENCE ENUMERATORS

Linear finite-state machines (LFSM's) form a very general class of finite-state machines with a variety of applications, including sequence enumerators, random number generators, encoders and decoders for linear error-correcting codes, and cellular automata. The state evolution of an LFSM $\mathcal{S}$ is given by

$$\mathbf{q}_s[t+1] = \mathbf{A}\mathbf{q}_s[t] \oplus \mathbf{B}\mathbf{x}[t] \ , \qquad (1)$$

where t is the discrete-time index, $\mathbf{q}_s[t]$ is the *state vector* and $\mathbf{x}[t]$ is the *input vector*. We assume that $\mathbf{q}_s[\cdot]$ is d-dimensional, $\mathbf{x}[\cdot]$ is u-dimensional, and $\mathbf{A}$, $\mathbf{B}$ are constant matrices of appropriate dimensions. All vectors and matrices have entries in $GF(2)$, the Galois field of order 2, i.e., they are either "0" or "1". Matrix-vector multiplication and vector-vector addition are performed as usual except that element-wise addition and multiplication are taken modulo-2. Operation $\oplus$ in (1) denotes vector addition modulo-2.

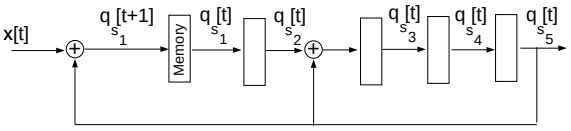

Fig. 2: Example of a sequence enumerator.

One can obtain an LFSM $\mathcal{S}'$ (with d-dimensional state vector $\mathbf{q}'_s[t]$) that is *similar* to $\mathcal{S}$ in eq. (1) through a similarity transformation, [10]:

$$\mathbf{q}'_s[t+1] \ = \ \underbrace{(\mathbf{T}^{-1}\mathbf{A}\mathbf{T})}_{\mathbf{A}'}\mathbf{q}'_s[t] \oplus \underbrace{(\mathbf{T}^{-1}\mathbf{B})}_{\mathbf{B}'}\mathbf{x}[t] \ ,$$

where $\mathbf{T}$ is an *invertible* $d \times d$ binary matrix such that $\mathbf{q}_s[t] = \mathbf{T}\mathbf{q}'_s[t]$. The initial conditions for the transformed LFSM can be obtained as $\mathbf{q}'_s[0] = \mathbf{T}^{-1}\mathbf{q}_s[0]$.

The linear feedback shift register (LFSR) in Fig. 2 is an example of a sequence enumerator. It is implemented using single-bit memory elements (flip-flops) and 2-input XOR gates (a 2-input XOR gate performs modulo-2 addition on its binary inputs and is denoted by $\oplus$ in the figure). The state ("current count") of the enumerator is given by the values stored in the single-bit memory elements; the corresponding state evolution equation can be written as the state evolution of an LFSM

$$\begin{aligned}\mathbf{q}_s[t+1] &= \mathbf{A}\mathbf{q}_s[t] \oplus \mathbf{b}x[t] \\ &= \begin{bmatrix} 0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,0 \end{bmatrix} \mathbf{q}_s[t] \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} x[t] \ . \qquad (2)\end{aligned}$$

Note that when $x[\cdot] = 0$ and $\mathbf{q}_s[0] \neq \mathbf{0}$, the LFSR acts as an autonomous *sequence enumerator* and goes through all non-zero states (essentially counting from 1 to 31). For example, if initialized at

$$\mathbf{q}_s[0] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T \ ,$$

the LFSR goes through states $\mathbf{q}_s[1] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$, $\mathbf{q}_s[2] = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$, ..., $\mathbf{q}_s[30] = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix}^T$, $\mathbf{q}_s[31] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$. The reason for this behavior is that the *feedback polynomial* for the shift register has been chosen to be primitive, [11].

Given the state evolution description of an LFSM (as in eq. (1) or eq. (2)), there are a number of possible implementations (using 2-input XOR gates and flip-flops). The situation is similar to the case of linear time-invariant dynamic systems [5, 6], and the case of state variable descriptions, [12]. Here, we assume an implementation where each bit in the next-state vector

$\mathbf{q}_s[t+1]$ is calculated using a *separate* set of 2-input XOR gates (i.e., no hardware is shared, as is the case in Fig. 2). This implies that a failure in a single XOR gate can corrupt at most one bit in the next-state vector $\mathbf{q}_s[t+1]$. We also assume that the calculation of each bit in $\mathbf{q}_s[t+1]$ is based on the bits of $\mathbf{q}_s[t]$ that are explicitly specified by the "1's" in matrix $\mathbf{A}$ of the state evolution equation (e.g., the third bit of $\mathbf{q}_s[t+1]$ in the LFSR of Fig. 2 is calculated based on the second and fifth bits of $\mathbf{q}_s[t]$). Under these assumptions, a single failure in an XOR gate corrupts at most one bit in the state vector of our redundant implementation; therefore, we can focus on detecting/correcting single-bit errors.

In order to protect a given LFSM $\mathcal{S}$ (with d state variables and state evolution as in eq. (1)) against corruptions of bits in its state vector, we embed it into a redundant LFSM $\mathcal{H}$ with η state variables ($\eta \equiv d+s$, $s>0$) and state evolution

$$\mathbf{q}_h[t+1] = \mathcal{A}\mathbf{q}_h[t] \oplus \mathcal{B}\mathbf{x}[t] \ . \qquad (3)$$

The initial state $\mathbf{q}_h[0]$ and matrices $\mathcal{A}$, $\mathcal{B}$ are chosen so that the state $\mathbf{q}_h[t]$ of $\mathcal{H}$ at time step t provides complete information about $\mathbf{q}_s[t]$, the state of the original LFSM $\mathcal{S}$, through a decoding mapping, and vice-versa. More specifically, we will restrict ourselves to decoding and encoding techniques that are linear in $GF(2)$; i.e., we assume that there exist

- a $d \times \eta$ binary *decoding* matrix $\mathbf{L}$ such that $\mathbf{q}_s[t] = \mathbf{L}\mathbf{q}_h[t]$ for all t, and
- an $\eta \times d$ binary *encoding* matrix $\mathbf{G}$ such that $\mathbf{q}_h[t] = \mathbf{G}\mathbf{q}_s[t]$ for all t.

Under the above assumptions, the redundant machine $\mathcal{H}$ concurrently simulates the original machine $\mathcal{S}$ ($\mathbf{q}_s[t] = \mathbf{L}\mathbf{q}_h[t]$). Furthermore, there is a one-to-one correspondence between the states in $\mathcal{S}$ and the states in $\mathcal{H}$ (i.e., $\mathbf{q}_h[t] = \mathbf{G}\mathbf{q}_s[t]$ *and* $\mathbf{q}_s[t] = \mathbf{L}\mathbf{q}_h[t]$).

Clearly, the redundant machine $\mathcal{H}$ enforces an (η, d) *linear code* on the state of the original machine. An (η, d) linear code uses η bits to represent d bits of information and is defined in $GF(2)$ by an $\eta \times d$ generator matrix $\mathbf{G}$ with full-column rank, [13, 11]. The d dimensional vector $\mathbf{q}_s[\cdot]$ is *uniquely* represented by the η dimensional vector (*codeword*) $\mathbf{q}_h[\cdot] = \mathbf{G}\mathbf{q}_s[\cdot]$. Error-detection is straightforward: under fault-free conditions, the redundant state vector must be in the column space of $\mathbf{G}$; therefore, all we need to check is that at each time step t the redundant state $\mathbf{q}_h[t]$ lies in the column space of $\mathbf{G}$ (in coding theory terminology we need to check that $\mathbf{q}_h[t]$ is a codeword of the linear code generated by $\mathbf{G}$, [13, 11]). Equivalently, we can check that $\mathbf{q}_h[t]$ is in the null space of an appropriate *parity check matrix* $\mathbf{P}$, so that $\mathbf{P}\mathbf{q}_h[t] = \mathbf{0}$. The parity check matrix has row rank $\eta - d \equiv s$ and satisfies $\mathbf{P}\mathbf{G} = \mathbf{0}$. Error-correction associates with each valid state in $\mathcal{H}$ (of the form $\mathbf{G}\mathbf{q}_s[\cdot]$), a unique subset of invalid states that get corrected to that particular valid state[3]. Since a single failure in an XOR gate results in the corruption of a single bit, error-correction can be performed using any of the methods used in the communications setting, [13, 11].

Theorem 2.1 *In the setting described above, LFSM $\mathcal{H}$ (of dimension $\eta \equiv d+s$, $s>0$ and state evolution as in eq. (3)) is a redundant version of $\mathcal{S}$ if and only if it is* similar *to a* standard *redundant LFSM $\mathcal{H}_\sigma$ whose state evolution equation is given by*

$$\mathbf{q}_\sigma[t+1] = \begin{bmatrix} \mathbf{A} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix} \mathbf{q}_\sigma[t] \oplus \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \mathbf{x}[t] \ . \quad (4)$$

Here, $\mathbf{A}$ and $\mathbf{B}$ are the matrices in eq. (1), $\mathbf{A}_{22}$ is an $s \times s$ binary matrix that describes the dynamics of the redundant modes that have been added, and $\mathbf{A}_{12}$ is a $d \times s$ binary matrix that describes the coupling from the redundant to the non-redundant modes. Associated with this standard redundant LFSM is the standard decoding matrix $\mathbf{L}_\sigma = \begin{bmatrix} \mathbf{I}_d & \mathbf{0} \end{bmatrix}$, the standard encoding matrix $\mathbf{G}_\sigma = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{0} \end{bmatrix}$ and the standard parity check matrix $\mathbf{P}_\sigma = \begin{bmatrix} \mathbf{0} & \mathbf{I}_s \end{bmatrix}$.

Proof: The proof of the theorem is discussed in [5] (it follows similar steps as the proof of the theorem in [6]). □

Given a pair of encoding and decoding matrices $\mathbf{L}$ and $\mathbf{G}$ (they need to satisfy $\mathbf{L}\mathbf{G} = \mathbf{I}_d$), and an LFSM $\mathcal{S}$, Theorem 2.1 completely characterizes all possible redundant LFSM's $\mathcal{H}$. Since the choice of the binary matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{22}$ is completely free, there are multiple redundant implementations of LFSM $\mathcal{S}$ for the given $\mathbf{L}$ and $\mathbf{G}$.

3. FAULT-TOLERANT SEQUENCE ENUMERATORS

In this section we discuss fault tolerance in sequence enumerators that are implemented as LFSM's. As we

[3]This subset usually contains η-bit vectors with small *Hamming* distance from the associated valid codeword. The Hamming distance between two binary vectors $x = (x_1, x_2, ..., x_\eta)$ and $y = (y_1, y_2, ..., y_\eta)$ is the number of positions at which x and y differ, [13, 11]. The minimum Hamming distance d_{min} of a code (collection of binary vectors of length η) determines its error detecting and correcting capabilities: a code can detect $d_{min}-1$ single-bit errors; it can correct $\lfloor \frac{d_{min}-1}{2} \rfloor$ single-bit errors.

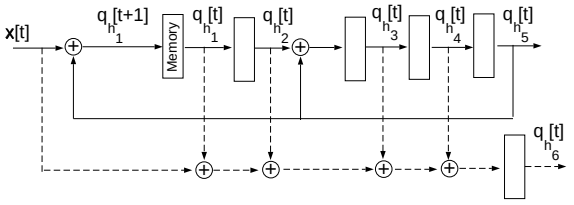

Fig. 3: Redundant implementation based on a checksum condition.

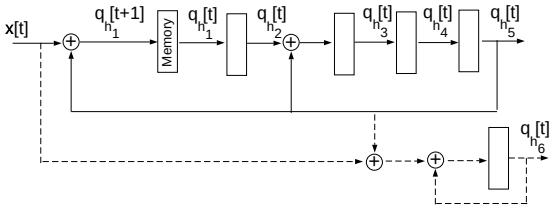

Fig. 4: Second redundant implementation based on a checksum condition.

illustrate via an example, choosing $\mathbf{A}_{12}$ and/or $\mathbf{A}_{22}$ appropriately can lead to designs that make better use of redundancy. Additional examples can be found in [5].

Suppose that the sequence enumerator $\mathcal{S}$ that we would like to protect is the linear feedback shift register shown in Fig. 2. In order to detect a single failure in an XOR gate, we can use an extra "checksum" state variable (as was suggested for linear time-invariant dynamic systems in [2] and for LFSM's in [14, 15]). The resulting redundant LFSM $\mathcal{H}$ has six state variables and state evolution

$$\mathbf{q}_h[t+1] = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{c^T A} & \mathbf{0} \end{bmatrix} \mathbf{q}_h[t] \oplus \begin{bmatrix} \mathbf{b} \\ \mathbf{c}^T\mathbf{b} \end{bmatrix} x[t]$$

$$= \left[\begin{array}{ccccc|c} 0&0&0&0&1&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&1&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ \hline 1&1&1&1&0&0 \end{array}\right] \mathbf{q}_h[t] \oplus \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 1 \end{array}\right] x[t] \,,$$

where $\mathbf{c}^T = \left[\begin{array}{ccccc} 1&1&1&1&1 \end{array}\right]$. A hardware implementation based on 2-input XOR gates and flip-flops is shown in Fig. 3. Under fault-free conditions, the added state variable $q_{h_6}[t]$ is the modulo-2 sum of all other state variables (which are the same as the original state variables in LFSM $\mathcal{S}$).

The above approach is easily seen to be consistent with our setup with encoding, decoding and parity check matrices given as

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_5 \\ \mathbf{c}^T \end{bmatrix} \,,\; \mathbf{L} = \left[\begin{array}{c|c} \mathbf{I}_5 & \mathbf{0} \end{array}\right] \,,\; \mathbf{P} = \left[\begin{array}{c|c} \mathbf{c}^T & 1 \end{array}\right] \,.$$

Using the transformation $\mathbf{q}_\sigma[t] = \mathcal{T}\mathbf{q}_h[t]$ where $\mathcal{T} = \begin{bmatrix} \mathbf{I}_5 & \mathbf{0} \\ \mathbf{c}^T & 1 \end{bmatrix}$, we see that $\mathcal{H}$ is similar to a standard redundant LFSM $\mathcal{H}_\sigma$ with state evolution

$$\mathbf{q}_\sigma[t+1] = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \mathbf{q}_\sigma[t] \oplus \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix} x[t] \,.$$

Note that both $\mathbf{A}_{12}$ and $\mathbf{A}_{22}$ are set to zero.

As stated earlier, there are multiple redundant implementations with the same encoding, decoding and parity check matrices. For the scenario described here, there are exactly 2^5 different LFSM's (we get a different system for each combination of choices for entries in matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{22}$). If we choose

$$\mathbf{A}_{12} = \mathbf{0} \,, \qquad \mathbf{A}_{22} = [1] \,,$$

then the same transformation ($\mathbf{q}_\sigma[t] = \mathcal{T}\mathbf{q}_{h'}[t]$, $\mathcal{T} = \begin{bmatrix} \mathbf{I}_5 & \mathbf{0} \\ \mathbf{c}^T & 1 \end{bmatrix}$) results in a redundant LFSM $\mathcal{H}'$ with state evolution equation

$$\mathbf{q}_{h'}[t+1] = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{c}^T\mathbf{A} - \mathbf{A}_{22}\mathbf{c}^T & \mathbf{A}_{22} \end{bmatrix} \mathbf{q}_{h'}[t] \oplus$$

$$\oplus \begin{bmatrix} \mathbf{b} \\ \mathbf{c}^T\mathbf{b} \end{bmatrix} x[t] =$$

$$= \left[\begin{array}{ccccc|c} 0&0&0&0&1&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&1&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ \hline 0&0&0&0&1&1 \end{array}\right] \mathbf{q}_{h'}[t] \oplus \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 1 \end{array}\right] x[t] \,.$$

A hardware implementation based on 2-input XOR gates and flip-flops is shown in Fig. 4.

Both redundant sequence enumerators $\mathcal{H}$ and $\mathcal{H}'$ have the same encoding, decoding and parity check matrices, and allow concurrent detection of single-bit errors in the redundant state vector (and therefore, according to our assumptions about hardware implementation, allow concurrent detection of a failure in a single XOR gate). Clearly, the complexity in $\mathcal{H}'$ is lower than in $\mathcal{H}$. The gain is not only in terms of the hardware involved, but also in terms of minimizing the probability of failure (since XOR gates may fail).

Note that one of the problems in encoding the state of dynamic systems (in order to provide fault tolerance) has been the cost associated with generating the redundant bits, [16]. For example, in the original implementation $\mathcal{H}$ of the checksum scheme, generating one additional bit costs more (in terms of 2-input XOR gates) than the linear feedback shift register altogether. As illustrated in this example for the case of a non-zero $\mathbf{A}_{22}$, we can obtain more efficient redundant imple-

mentations by exploiting the dynamics of the redundant modes (given by $\mathbf{A}_{22}$) and/or their coupling with the original system (given by $\mathbf{A}_{12}$). In [5], we discuss how to systematically choose $\mathbf{A}_{12}$ and $\mathbf{A}_{22}$ so that we minimize the complexity (number of XOR gates) of the fault-tolerant implementation.

4. SUMMARY

In this paper we studied a general approach for designing fault-tolerant sequence enumerators by encoding the states and dynamics of the corresponding LFSM's. The added redundancy can be used to protect against hardware, software or communication failures. We presented a variety of possible redundant implementations and described ways to minimize the hardware cost and/or overall complexity by exploiting non-zero redundant dynamics and coupling. The discussion in this paper assumed fault-free error-correcting mechanisms; we relax this assumption in [5, 17], where we use a related approach to construct fault-tolerant LFSM's largely out of unreliable components (unreliable XOR gates and voters). Future work should better characterize the tradeoffs involved among the different redundant implementations, particularly under non-separate linear codes. We should also study whether the two-stage approach to fault tolerance can employ convolutional rather than block coding techniques.

5. REFERENCES

[1] T. R. N. Rao, *Error Coding for Arithmetic Processors*. New York: Academic Press, 1974.

[2] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Transactions on Computers*, vol. 33, pp. 518–528, June 1984.

[3] P. E. Beckmann, Fault-Tolerant Computation Using Algebraic Homomorphisms. PhD thesis, EECS Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

[4] C. N. Hadjicostis, "Fault-Tolerant Computation in Semigroups and Semirings," M. Eng. thesis, EECS Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995.

[5] C. N. Hadjicostis, Coding Approaches to Fault Tolerance in Dynamic Systems. PhD thesis, EECS Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1999.

[6] C. N. Hadjicostis and G. C. Verghese, "Structured redundancy for fault tolerance in LTI state-space models and Petri nets," *Kybernetika*, vol. 35, pp. 39–55, January 1999.

[7] C. N. Hadjicostis and G. C. Verghese, "Monitoring discrete event systems using Petri net embeddings," in *Application and Theory of Petri Nets 1999*, no. 1639 in Lecture Notes in Computer Science, pp. 188–208, 1999.

[8] R. Leveugle and G. Saucier, "Optimized synthesis of concurrently checked controllers," *IEEE Transactions on Computers*, vol. 39, pp. 419–425, April 1990.

[9] V. S. Iyengar and L. L. Kinney, "Concurrent fault detection in microprogrammed control units," *IEEE Transactions on Computers*, vol. 34, pp. 810–821, September 1985.

[10] T. L. Booth, *Sequential Machines and Automata Theory*. New York: Wiley, 1968.

[11] S. B. Wicker, *Error Control Systems*. Englewood Cliffs, New Jersey: Prentice Hall, 1995.

[12] R. A. Roberts and C. T. Mullis, *Digital Signal Processing*. Reading, Massachusetts: Addison-Wesley, 1987.

[13] R. E. Blahut, *Theory and Practice of Data Transmission Codes*. Reading, Massachusetts: Addison-Wesley, 1983.

[14] R. W. Larsen and I. S. Reed, "Redundancy by coding versus redundancy by replication for failure-tolerant sequential circuits," *IEEE Transactions on Computers*, vol. 21, pp. 130–137, February 1972.

[15] A. Sengupta, D. K. Chattopadhyay, A. Palit, A. K. Bandyopadhyay, and A. K. Choudhury, "Realization of fault-tolerant machines — linear code application," *IEEE Transactions on Computers*, vol. 30, pp. 237–240, March 1981.

[16] S. Niranjan and J. F. Frenzel, "A comparison of fault-tolerant state machine architectures for space-born electronics," *IEEE Transactions on Reliability*, vol. 45, pp. 109–113, March 1996.

[17] C. N. Hadjicostis and G. C. Verghese, "Fault-tolerant linear finite state machines," in *Proceedings of the 6th IEEE Int. Conf. on Electronics, Circuits and Systems*, vol. 2, pp. 1085–1088, September 1999.