

Fault Protection of Digital Sequential Systems using Convolutional Codes over Large Alphabets*

Christoforos N. Hadjicostis[†] and Tim Ernst
Dept. of Electrical and Computer Eng. and Coordinated Science Lab.
University of Illinois
Urbana, IL 61801-2307, USA
{chadjic, ternst}@uiuc.edu

July 13, 2004

Abstract

In digital sequential systems that operate over several time steps, a state-transition fault at *any* time step during the operation of the system can corrupt its state in a way that renders its future functionality useless. In this paper, we discuss a methodology for systematically constructing redundant sequential systems whose state is encoded in a way that allows an external checker to capture transient state-transition faults via checks that are performed periodically. More specifically, this approach allows the checker to detect and identify errors due to *past* state-transition faults based on an analysis of the *current*, possibly corrupted system state. The motivation for periodic checking stems from our desire to relax the reliability requirements on the checker, which now operates at slower speeds than the original system. Our construction of redundant sequential systems makes use of coding techniques that are closely related to MDS convolutional codes over large alphabets.

1 Protection of Finite-State Machines

In this paper we further analyze embedding techniques that allow *nonconcurrent* error detection and identification in finite-state machines (FSMs) [1]. In these schemes, error-detection/identification is performed periodically instead of every time step, each time detecting and identifying errors due to transient state-transition faults that may have occurred since the last check. Our goal is to protect complicated computations in modern digital systems whose vulnerability to so-called “soft” (transient) faults has increased due to greater complexity, higher clock speeds, lower power consumption requirements and smaller transistor sizes. Our interest in nonconcurrent error detection, identification and correction stems from our desire to relax the reliability requirements on the error detector/corrector by allowing it to operate at slower speeds than the rest of the system.

Consider an FSM \mathcal{S} with state set $Q = \{q_1, q_2, \dots, q_N\}$ and input set $X = \{x_1, x_2, \dots, x_U\}$. Its state $q[t]$ and input $x[t]$ at time step t specify its next state $q[t+1]$ via the *next-state function*

$$q[t+1] = \delta(q[t], x[t]) . \quad (1)$$

We assume that function δ is defined for all pairs in $Q \times X$ and we are interested in detecting, identifying and correcting state-transition faults, i.e., faults that result in the corruption of the state of FSM \mathcal{S} at a particular time step.

*This material is based upon work supported in part by the National Science Foundation (under NSF Career Award 0092696 and NSF ITR Award 0218939) and in part by the Air Force Office of Scientific Research DoD (under AFOSR URI Award No F49620-01-1-0365URI). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF or AFOSR.

[†]Address for correspondence: University of Illinois at Urbana-Champaign, 148 CSL, 1308 West Main Street, Urbana, IL 61801-2307, USA.

In general, the implementation of FSM \mathcal{S} in Eq. (1) as a sequential system will rely on encoding each of its N states into a binary vector with b bits, where b is an integer satisfying $2^b \geq N$ (so that there are enough binary strings to represent all N states). One usually focuses on the case where the state encodings are of minimal length (i.e., $2^{b-1} \leq N \leq 2^b$ or $b = \lceil \log_2 N \rceil$), but our discussion can be easily generalized to arbitrary $b \geq \lceil \log_2 N \rceil$. Clearly, the states of the FSM \mathcal{S} can be captured by a set of N b -dimensional binary vectors

$$\mathcal{Q}_s = \{\mathbf{q}_s^{(1)}, \mathbf{q}_s^{(2)}, \dots, \mathbf{q}_s^{(N)}\}$$

that correspond to states $\{q_1, q_2, \dots, q_N\}$ respectively. These vectors can certainly be viewed as b -dimensional vectors with elements in $GF(2)$. One can also view \mathcal{Q}_s as a set of lower dimensional vectors with elements in a higher dimensional field. More generally, we can think of \mathcal{Q}_s as d -dimensional vectors with entries in $GF(q)$ where $d \geq \lceil \log_q N \rceil$ with $q \geq 2$. As we will see, the breakdown of the vector dimensions and the size of the finite field turns out to be an important factor in the complexity of the redundant implementation and the error-detection/identification mechanism.

For notational convenience, we define the following state transition mappings: for each input $x_u \in X_U$, let $\delta_{x_u} : \mathcal{Q}_s \mapsto \mathcal{Q}_s$ be the mapping that defines the state transition functionality under input x_u , i.e., $\delta_{x_u}(\mathbf{q}_s^{(j)}) = \mathbf{q}_s^{(l)}$ iff $q_l = \delta(q_j, x_u)$. With this notation

$$\mathbf{q}[t+1] = \delta_{x[t]}(\mathbf{q}[t]), \quad (2)$$

where $x[t] \in X_U$ is the input at time step t , and $\mathbf{q}[t], \mathbf{q}[t+1] \in \mathcal{Q}_s$ are the machine states at time steps t and $t+1$.

2 Embeddings of FSMs

To protect against transient state-transition faults in a given FSM, [1] constructs a larger FSM \mathcal{H} with an η -dimensional state vector $\xi[t]$ with entries in $GF(q)$, where $\eta = d + m$ for some $m > 0$. Using the notation introduced in the previous section, the state evolution of \mathcal{H} can be described by

$$\xi[t+1] = \Delta_{x[t]}(\xi[t]), \quad (3)$$

where $x[t] \in X_U$ is the input at time step t , and $\xi[t], \xi[t+1] \in \mathcal{Q}_h$ are the machine states at time steps t and $t+1$ [\mathcal{Q}_h is the η -dimensional vector space with entries in $GF(q)$]. The initial state $\xi[0]$ and all transition mappings $\Delta_{x_u}(\cdot)$, $x_u \in X$, are chosen in [1] so that under fault-free conditions the state $\xi[t]$ of FSM \mathcal{H} for $t \geq 0$ provides complete information about $\mathbf{q}[t]$, the state of the original FSM \mathcal{S} , and vice-versa. More specifically, we restrict ourselves to decoding and encoding mappings that are linear in $GF(q)$, i.e., we require that there exist a $d \times \eta$ matrix \mathbf{L} and an $\eta \times d$ matrix \mathbf{G} [with entries in $GF(q)$] such that, when $\xi[0] = \mathbf{G}\mathbf{q}[0]$ and under fault-free conditions,

$$\mathbf{q}[t] = \mathbf{L}\xi[t], \quad (4)$$

$$\xi[t] = \mathbf{G}\mathbf{q}[t] \quad (5)$$

for all $t \geq 0$. The second constraint implies that error detection, when performed concurrently, can be based on simply verifying whether the *syndrome* $\mathbf{s}[t] \equiv \mathbf{H}\xi[t]$ is nonzero (here, \mathbf{H} is the $m \times \eta$ parity check matrix, i.e., a matrix that has full row-rank and satisfies $\mathbf{H}\mathbf{G} = \mathbf{0}$). Concurrent error correction can be achieved by associating each valid state in FSM \mathcal{H} (of the form $\mathbf{G}\mathbf{q}[\cdot]$) with a unique subset of invalid states that get corrected to that particular valid state. The FSM \mathcal{H} defined in Eq. (3) that satisfies the constraints in Eqs. (4) and (5) if properly initialized is called a *redundant implementation* for FSM \mathcal{S} in Eq. (2).

In this paper we study a class of redundant FSM implementations that were defined in [1]. The proof of the following theorem can be found in [1].

Theorem 2.1 *Let the decoding, encoding and parity check matrices of a redundant implementation \mathcal{H} for FSM \mathcal{S} be given by*

$$\mathbf{L} = \begin{bmatrix} \mathbf{I}_d & \mathbf{0} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{C} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} -\mathbf{C} & \mathbf{I}_m \end{bmatrix}.$$

FSM \mathcal{H} is a redundant implementation for FSM \mathcal{S} [i.e., the state of FSM \mathcal{H} satisfies the decoding and encoding constraints in Eqs. (4) and (5)] if it is initialized in state $\xi[0] = \mathbf{G}\mathbf{q}[0]$ and its next-state transition mapping under each input $x_u \in X_U$ satisfies

$$\Delta_{x_u}(\xi[t]) = \left[\frac{\delta_{x_u}(\mathbf{q}[t]) \oplus \mathbf{A}_{12_{x_u}} \mathbf{C}\mathbf{q}[t] \oplus \mathbf{A}_{12_{x_u}} \xi_r[t]}{\mathbf{C}\delta_{x_u}(\mathbf{q}[t]) \oplus (\mathbf{C}\mathbf{A}_{12_{x_u}} \mathbf{C} \oplus \mathbf{A}_{22_{x_u}} \mathbf{C})\mathbf{q}[t] \oplus (\mathbf{C}\mathbf{A}_{12_{x_u}} \oplus \mathbf{A}_{22_{x_u}})\xi_r[t]} \right], \quad (6)$$

where $\xi[t] \equiv \begin{bmatrix} \mathbf{q}[t] \\ \xi_r[t] \end{bmatrix} = \mathbf{G}\mathbf{q}[t]$ under fault-free conditions [here $\mathbf{q}[t]$ is a d -dimensional vector (in \mathcal{Q}_s), $\xi_r[t]$ is an m -dimensional vector, and $\mathbf{A}_{12_{x_u}}$ and $\mathbf{A}_{22_{x_u}}$ are matrices of appropriate dimensions, all with entries in $GF(q)$].

Note that the coupling matrices $\mathbf{A}_{12_{x_u}}$, $x_u \in X_U$, and the redundant dynamics matrices $\mathbf{A}_{22_{x_u}}$, $x_u \in X_U$, are free for us to choose. Also note that the influence of the coupling and the redundant dynamics on the overall FSM state is restricted to be linear.

3 Transient State-Transition Faults and Error Propagation

We focus on transient faults, i.e., faults that do not appear on a consistent basis but only manifest themselves with a certain probability (transient faults are usually caused by glitches due to noise, electromagnetic interference, or environmental factors [2]). Since transient state-transition faults are actually harder to deal with than transient faults that corrupt the output of the FSM, we focus on protecting FSMs against transient *state-transition* faults. We assume that a transient fault during the calculation of the next state at time step t causes an erroneous value at exactly one of the state variables in the next state vector $\xi[t+1]$ (in VLSI implementations of digital sequential systems, this will be true if, for example, FSM \mathcal{H} is implemented so that the next value of each state variable is calculated by separate hardware [3]). This restriction is not critical and can easily be relaxed, but is taken into account when comparing costs of different redundant implementations in later sections of this paper.

To analyze nonconcurrent error detection and identification, we assume (without loss of generality) that the FSM begins operation with $\xi[0] = \mathbf{G}\mathbf{q}[0]$, and that the first nonconcurrent parity check is performed at the end of time step $L-1$ (i.e., at the beginning of time step L). We assume that a total of D transient state-transition faults occur during the executions of time steps $L-k_1-1, L-k_2-1, \dots, L-k_D-1$, originally corrupting state variables i_1, i_2, \dots, i_D by initial additive errors v_1, v_2, \dots, v_D , respectively. Without loss of generality, we assume that $0 \leq k_D \leq k_{D-1} \leq \dots \leq k_2 \leq k_1 \leq L-1$. The following theorem is taken from [1].

Theorem 3.1 *Let the redundant implementation \mathcal{H} for FSM \mathcal{S} be constructed as in Theorem 2.1. Assume that a total of D transient faults take place in the interval $[0, L-1]$. More specifically, transient faults occur during the executions of time steps $0 \leq L-k_1-1 \leq L-k_2-1 \leq \dots \leq L-k_D-1 \leq L-1$, originally corrupting state variables i_1, i_2, \dots, i_D by initial additive errors v_1, v_2, \dots, v_D . Then, the nonconcurrent syndrome $\mathbf{s}[L]$ is given by*

$$\mathbf{s}[L] \equiv \mathbf{H}\xi_f[L] = \sum_{j=1}^D \left\{ v_j \left(\prod_{\tau=L-k_j}^{L-1} \mathbf{A}_{22_{x[\tau]}} \right) \mathbf{H}\mathbf{e}_{i_j} \right\}.$$

Clearly, if redundant FSMs are constructed so that each FSM input is associated with the same redundant dynamics (as given by a matrix \mathbf{A}_{22}), then the syndrome $\mathbf{s}[L] \equiv \mathbf{H}\xi_f[L]$ at the end of time step $L-1$ (beginning of time step L) will be

$$\mathbf{s}[L] \equiv \mathbf{H}\xi_f[L] = \sum_{j=1}^D v_j \mathbf{A}_{22}^{k_j} \mathbf{H}\mathbf{e}_{i_j}, \quad (7)$$

i.e., the syndrome will be a linear combination of D columns of the $m \times L\eta$ syndrome matrix

$$\mathbf{S} = \left[\mathbf{H} \quad \mathbf{A}_{22}\mathbf{H} \quad \mathbf{A}_{22}^2\mathbf{H} \quad \dots \quad \mathbf{A}_{22}^{L-1}\mathbf{H} \right]. \quad (8)$$

In order to be able to detect D or less errors in the interval $[0, L-1]$ based on the syndrome $\mathbf{s}[L]$, we need all linear combinations of any subset of D columns of \mathbf{S} to be nonzero; to be able to uniquely identify the originally affected

variables (i_1, i_2, \dots, i_D) , the values by which they were initially corrupted (v_1, v_2, \dots, v_D) and the time steps during which the errors took place $(L - k_1 - 1, L - k_2 - 1, \dots, L - k_D - 1)$, we need all linear combinations of any subset of D columns of \mathbf{S} to be different from a linear combination of any other subset of D columns of \mathbf{S} . Notice that for the above class of redundant implementations (i.e., the class in which each FSM input is associated with the same redundant dynamics matrix \mathbf{A}_{22}), the $m \times L\eta$ syndrome matrix \mathbf{S} in Eq. (8) is the same as the one we had in [4] for nonconcurrent error detection and identification for LFSMs. Therefore, with appropriate choices of \mathbf{H} and \mathbf{A}_{22} , matrix \mathbf{S} can be chosen to be the parity check code of an MDS convolutional code [5, 6, 7].

The following theorem summarizes this observation. The proof is omitted. Note that in the construction below $\mathbf{V}(x_1, x_2, \dots, x_r)$ is defined as the $2D \times r$ matrix

$$\mathbf{V}(x_1, x_2, \dots, x_r) = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_r \\ x_1^2 & x_2^2 & \dots & x_r^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{2D-1} & x_2^{2D-1} & \dots & x_r^{2D-1} \end{bmatrix}$$

with elements in $GF(q)$.

Theorem 3.2 *Let the decoding, encoding and parity check matrices of a redundant implementation \mathcal{H} for FSM \mathcal{S} be given by*

$$\mathbf{L} = [\mathbf{I}_d \quad \mathbf{0}], \quad \mathbf{G} = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{C} \end{bmatrix}, \quad \mathbf{H} = [-\mathbf{C} \quad \mathbf{I}_m]$$

and assume that the next-state transition mapping under each input $x_u \in X_U$ satisfies

$$\Delta_{x_u}(\xi[t]) = \begin{bmatrix} \delta_{x_u}(\mathbf{q}[t]) \oplus \mathbf{A}_{12_{x_u}} \mathbf{C} \mathbf{q}[t] \oplus \mathbf{A}_{12_{x_u}} \xi_r[t] \\ \mathbf{C} \delta_{x_u}(\mathbf{q}[t]) \oplus (\mathbf{C} \mathbf{A}_{12_{x_u}} \mathbf{C} \oplus \mathbf{A}_{22} \mathbf{C}) \mathbf{q}[t] \oplus (\mathbf{C} \mathbf{A}_{12_{x_u}} \oplus \mathbf{A}_{22}) \xi_r[t] \end{bmatrix}$$

for all $\xi[t] = \begin{bmatrix} \mathbf{q}[t] \\ \xi_r[t] \end{bmatrix}$ [here $\mathbf{q}[t]$ is a d -dimensional vector (in \mathcal{Q}_s) and $\xi_r[t]$ is an m -dimensional vector, both with entries in $GF(q)$]. Any D or less errors due to transient state-transition faults in the interval $[0, L-1]$ can be detected and identified by a parity check at the end of time step $L-1$ (beginning of time step L) if the following conditions are satisfied:

- The number of additional state variables is $m = 2D$.
- The constants $x, x_1, x_2, \dots, x_\eta$ in the specified finite field $GF(q)$ are chosen so that

1. $x_i \neq x_j$ for $1 \leq i < j \leq \eta$;
2. $x^k x_i \neq x^{k'} x_j$ for integers k, k' such that $0 \leq k < k' \leq L-1$.

- The $2D \times 2D$ matrix \mathbf{A}_{22} is of the form

$$\mathbf{A}_{22} = \mathbf{M}^{-1} \mathbf{D} \mathbf{M},$$

where $\mathbf{D} = \text{diag}(1, x, x^2, x^3, \dots, x^{2D-1})$ and $\mathbf{M} = \mathbf{V}(x_{d+1}, x_{d+2}, \dots, x_\eta)$.

- The $2D \times d$ matrix \mathbf{C} is chosen so that

$$\mathbf{C} = -\mathbf{M}^{-1} \mathbf{V}(x_1, x_2, \dots, x_d).$$

As in the construction in [4], the order q of the finite field needs to be chosen to be large enough so that the first requirement can be satisfied. When the order of $GF(q)$ is small, one can still construct redundant FSMs that are amenable to nonconcurrent error detection and identification by using a greater number of additional state variables (this is shown in [6] in the context of BCH convolutional codes). If we modify the choice of matrices in Theorem 3.2, we can

obtain redundant implementations that allow the use of the Peterson-Gorenstein-Zierler (PGZ) algorithm to efficiently determine the D' errors based on the syndrome $\mathbf{s}[L]$ without resorting to an exhaustive search.

As expected, the size of the finite field q turns out to be extremely important in determining the complexity of a redundant implementation for FSM \mathcal{S} . In general, the *overhead* associated with a particular design for a certain D and L appears to go down with the size of the original implementation, i.e., the overhead goes down with d . The next section discusses an example and presents some of the insights obtained via experimental studies.

4 Example and Evaluation

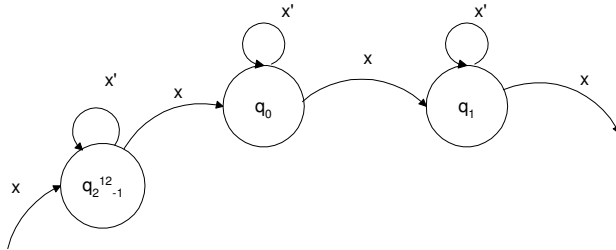


Figure 1: FSM counter with 2^{12} states.

4.1 Example

To illustrate the application of the proposed technique, we consider single error protection for the FSM in Figure 1. This is a 12-bit up-counter with enable: if the only input x is set to zero, the counter stays in the same state; if x is set to one the counter increments to the next state, eventually wrapping back to state zero. In terms of our earlier notation this FSM has two inputs, $x_1 = x$ and $x_2 = x'$. Its state can be encoded using 12 bits, e.g., by simply considering the binary value of each state: $0 \rightarrow 00 \dots 00$, $1 \rightarrow 00 \dots 01$, $2 \rightarrow 00 \dots 10$, etc. Note that one can obtain the next state logic that implements δ as a single logic block that receives 13 bits of input (12 bits for the current state and one bit for the input x) and produces 12 bits of output (that specify the next state). This logic block can be produced and optimized via numerous methods, however, the actual contents of δ are not necessary for our discussion here.

Note that the 12-bit state encoding can be treated as a 3-dimensional vector with elements in $GF(16)$. To ensure that a single fault corrupts a single variable we might want to produce separate logic for each one of them (i.e., split the logic that implements δ into three portions, each of which takes the same 13 bits of input and produces 4 bits of output¹). For the purposes of this example, we consider single error correction (double error detection) with the length of the check period being three cycles (i.e., $D = 1$ and $L = 3$). From Theorem 3.2, we know that single error correction requires a redundant state encoding with two additional state variables ($m = 2$). Since we are treating the 12-bit state encoding as a 3-dimensional vector in $GF(16)$, we have a total of five state variables and we need to choose constants $x, x_1, x_2, x_3, x_4, x_5$ in $GF(16)$ such that all $x^k x_i$ for $0 \leq k \leq 2$ and $1 \leq i \leq 5$ are distinct (Theorem 3.2). Clearly, we need at least $3 \times 5 = 15$ distinct nonzero numbers and $GF(16)$ is (barely) adequate for the purposes of our construction. For completeness, the elements of $GF(16)$ are shown in Table 1, including their power, polynomial vector and decimal representation. Choosing $x_1 = \alpha$, $x_2 = \alpha^2$, $x_3 = \alpha^3$, $x_4 = \alpha^4$, $x_5 = \alpha^5$ and $x = \alpha^5$ guarantees that the conditions of Theorem 3.2 are met for $L = 3$ (because each term will be a unique power of α with the last term ($x^2 x_5$) being $\alpha^{15} = \alpha^0$).

Based on the above choice for constants $x, x_1, x_2, x_3, x_4, x_5$, the matrices \mathbf{A}_{22} and \mathbf{C} in Theorem 3.2 can be calculated

¹In other words, the logic blocks that produce the three state variables should not share any hardware.

Power	Polynomial	Vector	Regular	Power	Polynomial	Vector	Regular
0	0	0000	0	α^7	$\alpha^3 + \alpha + 1$	1011	11
α^0	1	0001	1	α^8	$\alpha^2 + 1$	0101	5
α^1	α	0010	2	α^9	$\alpha^3 + \alpha$	1010	10
α^2	α^2	0100	4	α^{10}	$\alpha^2 + \alpha + 1$	0111	7
α^3	α^3	1000	8	α^{11}	$\alpha^3 + \alpha^2 + \alpha$	1110	14
α^4	$\alpha + 1$	0011	3	α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$	1111	15
α^5	$\alpha^2 + \alpha$	0110	6	α^{13}	$\alpha^3 + \alpha^2 + 1$	1101	13
α^6	$\alpha^3 + \alpha^2$	1100	12	α^{14}	$\alpha^3 + 1$	1001	9

Table 1: Elements in $GF(16)$.

easily as follows:

$$\begin{aligned}
\mathbf{M} &= \mathbf{V}(x_4, x_5) = \begin{bmatrix} 1 & 1 \\ \alpha^4 & \alpha^5 \end{bmatrix}, \\
\mathbf{A}_{22} &= \mathbf{M}^{-1} \text{diag}(1, \alpha^5) \mathbf{M} \\
&= \begin{bmatrix} \alpha^{12} & \alpha^7 \\ \alpha^{11} & \alpha^7 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \alpha^5 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ \alpha^4 & \alpha^5 \end{bmatrix} \\
&= \begin{bmatrix} \alpha^{13} & \alpha^7 \\ \alpha^6 & \alpha^9 \end{bmatrix}, \\
\mathbf{C} &= -\mathbf{M}^{-1} \mathbf{V}(x_1, x_2, x_3) \\
&= -\begin{bmatrix} \alpha^{12} & \alpha^7 \\ \alpha^{11} & \alpha^7 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ \alpha & \alpha^2 & \alpha^3 \end{bmatrix} \\
&= \begin{bmatrix} \alpha^9 & \alpha^8 & \alpha^3 \\ \alpha^7 & \alpha^2 & \alpha^{14} \end{bmatrix}.
\end{aligned}$$

We can now easily describe the functionality of the redundant implementation: the original machine with next state transition functionality captured by

$$\mathbf{q}[t+1] = \delta_{x[t]}(\mathbf{q}[t])$$

(where \mathbf{q} is 12 bits wide and input $x[t]$ could be either zero or one) is replaced by the redundant machine with state evolution

$$\xi[t+1] = \Delta_{x[t]}(\xi[t]),$$

where

$$\xi[t] \equiv \begin{bmatrix} \mathbf{q}[t] \text{ (12 bits)} \\ \xi_r[t] \text{ (8 bits)} \end{bmatrix} = \begin{bmatrix} q_1[t] \text{ (4 bits)} \\ q_2[t] \text{ (4 bits)} \\ q_3[t] \text{ (4 bits)} \\ \xi_{r_1}[t] \text{ (4 bits)} \\ \xi_{r_2}[t] \text{ (4 bits)} \end{bmatrix}$$

and

$$\Delta_{x[t]} = \left[\frac{\delta_{x[t]}(\mathbf{q}[t])}{\mathbf{C} \delta_{x[t]}(\mathbf{q}[t]) \oplus \mathbf{A}_{22} \mathbf{C} \mathbf{q}[t] \oplus \mathbf{A}_{22} \xi_r[t]} \right] \quad (9)$$

(note that \mathbf{A}_{12} has been set to zero for simplicity).

To illustrate the operation of the redundant machine, consider a scenario where we start in state q_1 and input sequence $x = 1, x = 0, x = 1$ is applied so that under fault-free operation the machine would be traversing from state q_1 , to state q_2 , state q_2 and state q_3 . To be more precise, one should say that the redundant FSM starts in the state that *corresponds* to q_1 and traverses to the states that *correspond* to q_2 , q_2 and q_3 ; the exact fault-free state sequence is illustrated on the left of Table 2. Suppose that a fault during time step 2 corrupts the third state variable and causes a transition to

Time Step	FF State	FF State Vector	Faulty State	Faulty State Vector
0	1	$\{0, 0, \alpha^0 \alpha^3, \alpha^{14}\}$	-	-
1	2	$\{0, 0, \alpha^1 \alpha^4, \alpha^0\}$	-	-
2	2	$\{0, 0, \alpha^1 \alpha^4, \alpha^0\}$	13	$\{0, 0, \alpha^{13} \alpha^4, \alpha^0\}$
3	3	$\{0, 0, \alpha^4 \alpha^7, \alpha^3\}$	14	$\{0, 0, \alpha^{11} \alpha^6, \alpha^{13}\}$

Table 2: Example illustrating fault-free and faulty operation.

an erroneous state captured by the vector $\{0, 0, \alpha^{13} | \alpha^4, \alpha^0\}$ (instead of the state that corresponds to q_2). Note that the corruption of the third state variable from α^1 to α^{13} corresponds in this particular case to the corruption of all four bits associated with that variable.

During time step 3, the redundant machine evolves according to the input it receives: the part that corresponds to the state of the original machine traverses from state q_{13} to q_{14} , and the redundant part propagates the fault according to Eq. (9) as shown in Table 2. At the end of time step 3, when a check operation is performed, we obtain

$$\mathbf{s}[3] = \mathbf{H}\xi_r[3] = \begin{bmatrix} \alpha^8 \\ \alpha^9 \end{bmatrix}.$$

where $\mathbf{H} = [-\mathbf{C} \quad \mathbf{I}_2] = \begin{bmatrix} \alpha^9 & \alpha^8 & \alpha^3 & 1 & 0 \\ \alpha^7 & \alpha^2 & \alpha^{14} & 0 & 1 \end{bmatrix}$.

The syndrome matrix is constructed using Eq. (8) as

$$\mathbf{S} = [\mathbf{H} \mid \mathbf{A}_{22}\mathbf{H} \mid \mathbf{A}_{22}^2\mathbf{H}] = \left[\begin{array}{ccccc|ccccc|ccccc} \alpha^9 & \alpha^8 & \alpha^3 & 1 & 0 & \alpha^1 & \alpha^5 & \alpha^{11} & \alpha^{13} & \alpha^7 & \alpha^{10} & \alpha^6 & \alpha^{14} & \alpha^4 & \alpha^2 \\ \alpha^7 & \alpha^2 & \alpha^{14} & 0 & 1 & \alpha^4 & \alpha^{10} & \alpha^{12} & \alpha^6 & \alpha^9 & \alpha^5 & \alpha^{13} & \alpha^3 & \alpha^1 & \alpha^8 \end{array} \right].$$

In this case, it can be seen by inspection of \mathbf{S} that the syndrome vector is equal to α^{12} times the eighth column of \mathbf{S} . Using this information, we can now fill in the parameters of Eq. (7): since the eighth column of \mathbf{S} is the third column of the term $\mathbf{A}_{22}\mathbf{H}$, according to the notation of Eq. (7) this corresponds to $k_1 = 1$ and $i_1 = 3$ with $v_1 = \alpha^{12}$. Therefore, $\mathbf{s}[3]$ has been generated due to an additive error of α^{12} affecting the third variable during time step 2. This means the correct value of the third state variable at time step 2 should have been $\alpha^{13} \ominus \alpha^{12} = \alpha^1$, which is equal to the fault-free value shown in Table 2.

4.2 Implementation Overview

From our discussions in the previous sections, it is clear that there are a number of choices when implementing the redundant machine. Before evaluating this scheme, we first give an overview of the implementation. Recall that the update of the redundant part reduces (when \mathbf{A}_{12} is set to zero for all inputs) to

$$\xi_r[t+1] = \mathbf{C}\delta_x(\mathbf{q}[t]) \ominus \mathbf{A}_{22}\mathbf{C}\mathbf{q}[t] \oplus \mathbf{A}_{22}\xi_r[t].$$

There are two obvious approaches when implementing this equation: (i) treat $\xi_r[t+1]$ as the direct output of a Boolean function that takes as inputs $\mathbf{q}[t]$, $\xi_r[t]$ and $\mathbf{x}[t]$, or (ii) split $\xi_r[t+1]$ into a sequence of operations, taking advantage of the structure of matrix-vector multiplications. Here, we select the second approach because it leads to smaller overheads. The resulting implementation is shown in Figure 2 (notice that \mathbf{q} has been split into three variables and ξ_r has been split into two variables).

When splitting the calculation of the redundant part of the next state into three portions, everything but the $\mathbf{C}\delta$ portion can be performed as a series of multiplications and additions in $GF(q)$. The multipliers are implemented as logic blocks, whereas adders are either modulo- q adders (for a field of prime cardinality) or simply XOR gates (for an extension of $GF(2)$). The $\mathbf{C}\delta$ portion of the calculation cannot be broken up to prevent propagation of a single error to multiple variables in the redundant portion. As a final note, recall that the function δ must be altered to enforce the condition that each variable is calculated separately (so that a single error cannot propagate to multiple latches).

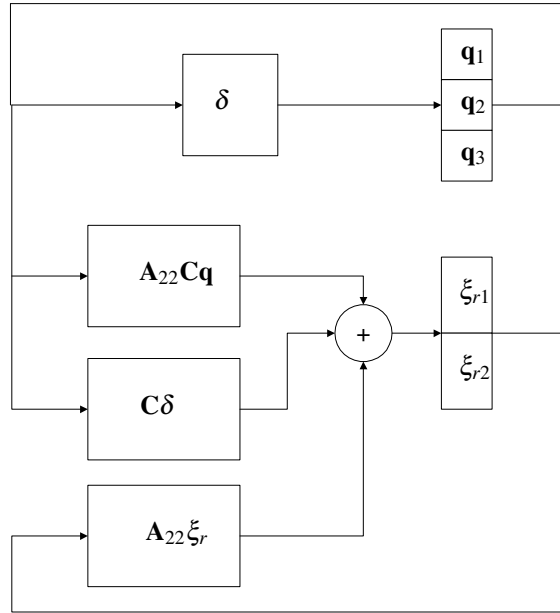


Figure 2: Block implementation of the redundant FSM.

4.3 Results

Figure 3 illustrates the overhead as a function of the size of the original finite-state machine for fields of prime cardinality as well as fields that are extensions of $GF(2)$. The data is for fixed q , \mathbf{C} , and \mathbf{A}_{22} to allow for better comparison. The FSMs used for comparison were randomly generated to be able to obtain an FSM with any number of states and representative complexity.² This graph shows two main trends: (i) the overhead decreases as the size of the FSM increases; and (ii) using a field that is an extension of $GF(2)$ results in significantly smaller overheads than using a field of prime cardinality. The general decrease in overhead as FSM size increases can be explained by the fact that the redundant calculation remains constant (in terms of the number of output bits), while the base FSM is increasing in complexity. This means that the next state logic from the original FSM will increase in size at a faster rate than that of the redundant portion (note that the logic in the redundant part is also increasing because the number of its primary inputs is increasing).

Fields that are extensions of $GF(2)$ appear to be superior to fields with prime cardinality, most likely due to the simplicity of addition and the $\mathbf{C}\delta$ calculation. For both types, simulations show that the $\mathbf{C}\delta$ portion of the calculation dominates the overhead of the redundant logic, especially as the machine size grows (Figure 4). This is not surprising because the size of the $\mathbf{A}_{22}\xi_r$ and $\mathbf{A}_{22}\mathbf{C}\mathbf{q}$ blocks is constant for fixed choices of \mathbf{A}_{22} and \mathbf{C} . Also, these blocks are simple to implement when one takes advantage of the structure of matrix-vector multiplication. In addition, the $\mathbf{C}\delta$ calculation appears to be simpler to implement in extension fields.

When q increases in number of bits, the overhead increases; this is expected since a larger q allows for a longer period between checks. However, as seen in Figure 5, for a field of prime cardinality, the relationship is non-monotonic.

²The FSM used in the example is very simple to implement and is not representative of the logic complexity associated with a typical FSM of its size.

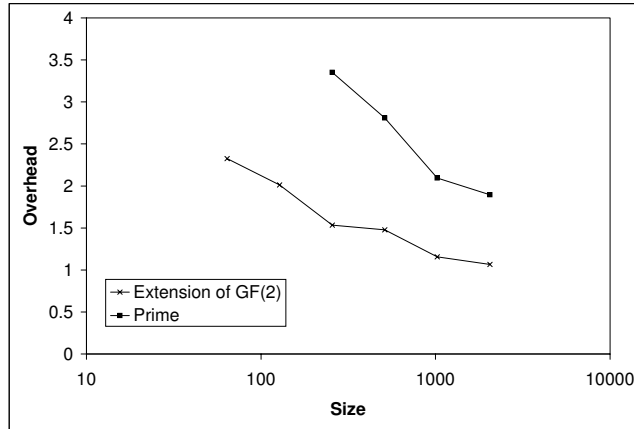


Figure 3: Overhead for various sizes of FSM for both a field of prime cardinality and field that is an extension field of $GF(2)$.

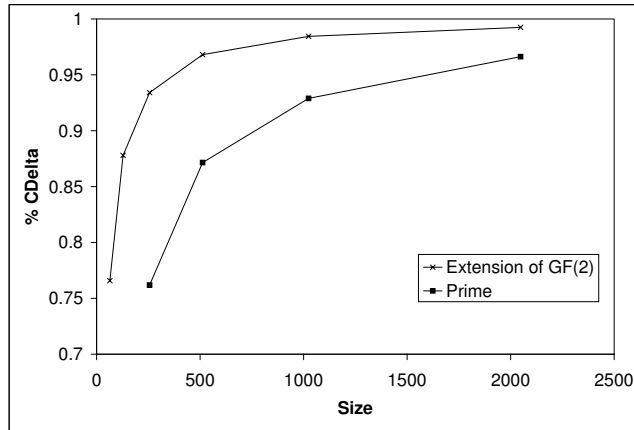


Figure 4: Percentage of overhead due to the $C\delta$ block.

5 CONCLUSIONS

The real challenge in extending nonconcurrent error detection and identification techniques to FSMs lies in our ability to add redundancy and effectively model transient state-transition faults in a way that allows us to capture the propagation of errors in the system. The schemes proposed in this paper are successful in systematically tracking errors, including errors in the added (redundant) variables. In particular, the encoded FSMs studied in this paper enable the use of well-known error detection and identification procedures. There are several parameters that need to be appropriately chosen before these techniques can be applied toward the construction of reliable sequential systems (i.e., window length L , additional state variables m , finite field size q , number of errors D that can be detected/identified). There are also a number of interesting practical questions: (i) How can the flexibility in the coupling between the redundant state variables and the original state variables [given by matrices $\mathbf{A}_{12 \times u}$ in Eq. (6)] be exploited to our advantage? (ii) How can we design systems that are optimal in terms of minimizing the redundant amount of hardware instead of minimizing the number of additional state variables? (iii) What other codes can be adopted to this setting and what is the corresponding choice of redundant dynamics? (iv) If one is only interested in k_1 (the earliest time step at which a fault took place, as

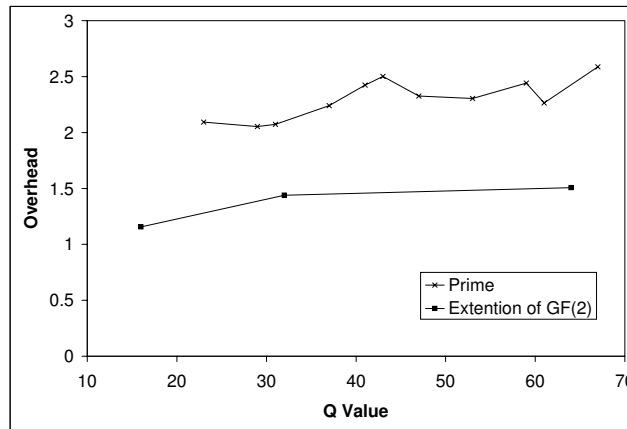


Figure 5: Overhead with varying values of q .

would be the case in rollback-based recovery), what type of codes and redundant implementations are appropriate? (v) What other connections can be made between algebraic systems theory and coding theory [8, 9] in terms of designing fault-tolerant FSMs? It would also be interesting to investigate these techniques in conjunction with methodologies for mapping algebraic equations to hardware.

References

- [1] C. N. Hadjicostis, "Finite-state machine embeddings for non-concurrent error detection and identification," in *Proceedings of CDC 2003, the 42nd IEEE Conf. on Decision and Control*, vol. 4, (Maui, Hawaii), pp. 3215–3220, December 2003.
- [2] D. Siewiorek and R. Swarz, *Reliable Computer Systems: Design and Evaluation*. Natick, Massachusetts: A.K. Peters, 1998.
- [3] C. N. Hadjicostis, *Coding Approaches to Fault Tolerance in Combinational and Dynamic Systems*. Boston, Massachusetts: Kluwer Academic Publishers, 2002.
- [4] C. N. Hadjicostis, "Non-concurrent error detection and correction in fault-tolerant linear finite state machines," *IEEE Transactions on Automatic Control*, vol. 48, pp. 2133–2140, December 2003.
- [5] J. Rosenthal, "Some interesting problems in systems theory which are of fundamental importance in coding theory," in *Proceedings of CDC 1997, the 36th Conf. on Decision and Control*, vol. 5, (San Diego, California), pp. 4574–4579, 1997.
- [6] J. Rosenthal and F. V. York, "BCH convolutional codes," *IEEE Transactions on Information Theory*, vol. 45, pp. 1833–1844, September 1999.
- [7] R. Smarandache, H. Gluesing-Luerssen, and J. Rosenthal, "Constructions of MDS-convolutional codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 2045–2049, July 2001.
- [8] B. Marcus and J. Rosenthal, eds., *Codes, Systems, and Graphical Models*, vol. 123 of *IMA Volumes in Mathematics and its Applications*. New York: Springer, 2001.
- [9] E. V. York, J. Rosenthal, and J. M. Schumacher, "On the relationship between algebraic systems theory and coding theory: representations of codes," in *Proceedings of CDC 1995, the 34th Conf. on Decision and Control*, vol. 3, (New Orleans, Louisiana), pp. 3271–3276, 1995.