Fault Tolerant Linear State Machines

Stefan Weidling and Michael Goessel
Department of Computer Science
University of Potsdam
Potsdam, Germany
Email: weidling@uni-potsdam.de, mgoessel@cs.uni-potsdam.de

Abstract—In this paper, a new method for the design of fault-tolerant linear state machines with initial state 0 and one-dimensional input and one-dimensional output is proposed. It is shown that the LFSR-implementation of the transfer function of a linear automaton can be utilized to correct transient errors in the memory elements. Since the state vector of a linear automaton is uniquely determined by the last \( n \) inputs and outputs, a transient error in a memory element can be corrected within \( n \) clock cycles by use of the corrected output symbols, where \( n \) is the number of components of the state vector. Experimental results have shown that the lowest area overhead can be obtained if the linear state machine is duplicated and a single parity bit is used to distinguish which of the duplicated machines is correct. In this area, an area overhead of 177\% for an 8-bit state vector and 160\% for a 256-bit state vector is achieved.

Keywords—fault tolerance; linear state machine; transient error; parity; transfer function

I. INTRODUCTION

Linear state machines [1], [2] are of importance in different fields of application such as encoding and decoding of cyclic codes, especially BCH codes [3], [4], generation of pseudo-random test patterns and test response compaction by multiple input signature analyzers [5], the implementation of stream ciphers [6] and others.

In modern technologies the number of errors in memory elements caused by transient faults is increasing and for many applications fault-tolerant designs are now of growing interest. The standard approach for fault tolerance is triple modular redundancy (TMR). A system is triplicated into three (functionally) identical systems and the outputs of the triplicated systems are connected to a three-input voter which determines the output of the fault-tolerant system as the majority of the outputs of the triplicated systems. The main advantages of TMR are that errors due to an arbitrary fault in only one of the triplicated systems are tolerated, and that no specific error model is required.

A very important disadvantage of this approach is that an error caused by a transient fault in a single memory element may result in an erroneous state of one of the triplicated systems for an infinite number of clock cycles. If in such a situation a second transient fault occurs in one of the other two systems, the overall system is not fault-tolerant anymore. As we will show, this will happen especially often in linear state machines.

In general an erroneous binary value of a state variable can be “corrected” without any additional hardware, when it is processed by a gate with a controlling value, as illustrated, for example, in Fig. 1.

In Fig. 1 the output of a first flip-flop \( FF_1 \) is connected via a first input of an \( AND \)-gate with the input of a second flip-flop \( FF_2 \). The state component \( z_1 \) of the flip-flop \( FF_1 \) is supposed to be disturbed at time \( t \) into an erroneous value \( z_1^e(t) \). The controlling value of the \( AND \)-gate is 0, and if the second input \( x(t) \) of the \( AND \)-gate \( x(t) = 0 \) is equal to the controlling value, the output \( x(t)z_1^e(t) \) of this \( AND \)-gate is equal to 0 independently of its erroneous input value \( z_1^e(t) \) at the first input line. The erroneous value \( z_1^e(t) \) of the erroneous state component \( z_1^e \) is corrected into 0. If \( x(t) = 1 \), which is the non-controlling value of an \( AND \)-gate, the erroneous state component \( z_1^e(t) \) is stored in the second flip-flop.

This simple example illustrates that an erroneous state component in one of the triplicated systems can be corrected by gates with a controlling value. If the controlling values and the non-controlling values of the \( AND \)-, \( NAND \)-, \( OR \)- and \( NOR \)-gates of the circuitry are applied with a certain probability, an erroneous state is corrected also with a certain probability.

But this is only possible if gates with controlling values are used for the design of the circuitry. If the system is a linear state machine implemented only by \( XOR \)-gates and memory elements, only the following two possibilities may occur:

- An error of the state of the linear state machine (caused either by a transient fault in the memory element itself or by a transient fault in the combinational part of the circuit) remains for an infinite number of clock cycles within the system or
- an error of the state of the linear state machine disappears with certainty after some clock cycles.

This is illustrated by the simple example of Fig. 2.

In Fig. 2 the circuitry is a linear state machine with a four-dimensional state vector \( z_1, z_2, z_3, z_4 \), one-dimensional input \( x \) and one-dimensional output \( y \). It is assumed that the first component \( z_1^e(t) \) of the state vector at time \( t \) is erroneous. Since the \( XOR \)-gate has no controlling value, at time \( t+1 \) the content of the second flip-flop \( FF_2 \) is \( z_2(t+1) = z_1^e(t) \oplus x(t) \neq z_1(t) \oplus x(t) \) which is erroneous. This erroneous value is stored at time \( t+2 \) in the flip-flop \( FF_3 \) and at time \( t+3 \) in the flip-flop \( FF_4 \). If there is a feedback connection from \( FF_4 \) to \( FF_1 \), the error caused by a transient fault at time \( t \) in the flip-flop \( FF_1 \) will never disappear from the overall system. If

---

Supported by Deutsche Forschungsgemeinschaft (DFG), under grant "GO 536/8-1".
with l-dimensional input and m-dimensional output for one-dimensional input and one-dimensional output. This method can be generalized to linear state machines with ordinary standard libraries are required for the design. For instance in [10], [12]–[16].

 approaches are possible. state encoding are compared. shown for instance in [7], [8]. In [9] the two fault tolerance also fault tolerance can be achieved by coding approaches as it is illustrated in Fig. 3 where the combinational circuit part “Gate” of a first system. No matter the gate is a gate with a controlling value or an XOR-gate with no controlling value. The main disadvantage of this approach is that it is costly.

Also fault tolerance can be achieved by coding approaches as shown for instance in [7], [8]. In [9] the two fault tolerance techniques Triple Modular Redundancy and Hamming-based state encoding are compared.

If not only standard libraries are available the following approaches are possible. Transient errors within the memory elements can also be tolerated by duplicated memory elements their outputs being connected to a C-element as proposed in [10], [11]. Also hardened memory elements can be used as it is described for instance in [10], [12]–[16].

In this paper we propose a new method for the design of fault-tolerant linear state machines with initial state zero. Only ordinary standard libraries are required for the design. For the simplicity of presentation we consider only linear state machines with one-dimensional input and one-dimensional output. This method can be generalized to linear state machines with l-dimensional input and m-dimensional output for l, m ≥ 1. For the considered automata the input-output behavior is completely determined by the transfer function which is the D-transformation of the impulse response. The proposed method utilizes the fact that the transfer function can be realized in its standard form by use of a multiple input linear feedback shift register (LFSR) with inputs x(t) and y(t). This realization allows us to implement fault tolerance in such a way that all errors in the state components of one of the registers will be automatically corrected in at most n clock cycles, where n is the number of components of the state vector. Although the system has feedback connections, this property can be achieved.

The rest of the paper is organized as follows. In section II basic properties of linear automata which are the theoretical model of linear state machines are given. The proposed method for the design of fault-tolerant linear state machines is described in section III. Two modifications are considered. In subsection III-A the multiple input LFSR implementing the transfer function of the considered linear state machine is triplicated, the outputs of the triplicated LFSR’s are processed by a three-input voter and the corrected output of the voter is fed back to the three LFSR’s. By this construction an arbitrary error in the state components of one of the LFSR’s is corrected within n cycles, where n is the number of components of the state vector. Also the voter can be triplicated. To reduce the necessary area for the implementation described in subsection III-A in subsection III-B the multiple input LFSR is only duplicated. Error detection by parity prediction is implemented for one or for both the states of the LFSR’s. The error-free output of one of the duplicated LFSR’s is fed back into both the duplicated LFSR’s. In section IV the proposed approaches are experimentally compared with the well-known solutions TMR, TMR for the flip-flops and state encoding by Hamming-codes. Paper achievements are summarized in section V.

II. BASIC PROPERTIES OF LINEAR STATE MACHINES

A. Definitions and Response formula

A linear state machine L which is also called a linear automaton over a field K is defined by the next-state function

\[ z(t+1) = A \cdot z(t) \oplus B \cdot x(t) \]

and the output function

\[ y(t) = C \cdot z(t) \oplus D \cdot x(t), \]

where \( x(t), y(t) \) and \( z(t) \) are the l-dimensional input vector, the m-dimensional output vector and the n-dimensional state vector at time \( t \) with its components from the field \( K \). Thereby \( t \) is a discrete time and \( \oplus \) is the addition in the field \( K \). The matrices \( A, B, C \) and \( D \) with elements from the field \( K \) are constant matrices of appropriate dimensions. For the

![Fig. 1: Correction of a state variable without any additional hardware.](image1)

![Fig. 2: Error cases of a linear state machine.](image2)

![Fig. 3: Triplication of the flip-flops and the combinational circuit part.](image3)
simplicity of presentation we assume here \( l = m = 1 \) and \( K = GF(2) \). Thus we consider binary linear automata with one-dimensional input and one-dimensional output. The addition \( \oplus \) is the addition modulo-2 and can be implemented by XOR-gates.

The state \( z(t) \) at time \( t \) of \( L \) is determined by

\[
z(t) = A^t \cdot z(0) + \sum_{i=0}^{t-1} A^{t-i} \cdot B \cdot x(i),
\]

where \( z(0) \) is the initial state and \( \{x\} = x(0), x(1), \ldots, x(t-1) \) is the input sequence. The output \( y(t) \) of \( L \) is given by

\[
y(t) = C \cdot A^t \cdot z(0) + \sum_{i=0}^{t} H(t-i) \cdot x(i)
\]

with

\[
H(t-i) = \begin{cases} D & \text{if } t-i = 0 \\ C \cdot A^{t-i} \cdot B & \text{if } t-1 - i \geq 0 \end{cases}
\]

Eq. (1) and Eq. (2) can be easily proved by induction [1].

If the initial state \( z(0) \) is the 0-state, the linear automaton is denoted by \( L_0 \) and the output \( y(t) \) of \( L_0 \) is determined by

\[
y(t) = \sum_{i=0}^{t} H(t-i) \cdot x(i).
\]

For an input sequence

\[
x(0), x(1), \ldots, x(t) = 1, 0, 0, \ldots, 0
\]

the corresponding output sequence, which is determined by Eq. (3), is

\[
y(0), y(1), \ldots, y(t) = H(0), H(1), \ldots, H(t).
\]

Thus if the input sequence \( 1, 0, 0, \ldots, 0 \) (which may be called an impulse sequence) is applied to a linear automaton \( L_0 \) with initial state \( z(0) = 0 \), the output sequence is \( H(0), H(1), \ldots, H(t) \). Therefore, \( H(t) \) is called the impulse response of the linear automaton \( L_0 \).

\[\text{Fig. 4: Realization of a linear automaton by its transfer function according to Eq. (4).}\]

\[\text{B. Transfer Function and its Implementation}\]

Equation (3), which describes the input-output behaviour of a linear automaton \( L_0 \) with initial state \( z(0) = 0 \), can be simplified by use of the D-transform. The D-transform of an infinite sequence \( \{x(t)\} = x(0), x(1), x(2), \ldots \) is a formal power series

\[
D[x(t)] = \sum_{i=0}^{\infty} x(i) \cdot d^i,
\]

where the exponent \( i \) of the formal variable \( d \) indicates the discrete time.

The D-transform of a finite sequence \( x(0), x(1), x(2), \ldots, x(T) \) of length \( T+1 \) is a polynomial of degree \( T \),

\[
D[x(0), x(1), x(2), \ldots, x(T)] = x(0) + x(1) \cdot d + x(2) \cdot d^2 + \ldots + x(T) \cdot d^T,
\]

and the D-transform of a periodic sequence \( \{x(t)\} = x(0), x(1), \ldots, x(r-1), x(r), x(r+1), \ldots, x(r+T-1) \) is

\[
D[x(t)] = \sum_{i=0}^{r-1} x(i) \cdot d^i + \frac{d^r}{1 - d} \sum_{j=0}^{T-1} x_{r+j} \cdot d^j.
\]

Now we assume that the considered linear automaton \( L_0 \) is finite. Then the impulse response is necessarily periodic and the transfer function, as the D-transform of a periodic function, can be brought into its normal form

\[
D[H(t)] = \beta_0 + \beta_1 \cdot d + \ldots + \beta_n \cdot d^n = \frac{\beta_0 + \beta_1 \cdot d + \ldots + \beta_n \cdot d^n}{1 + \alpha_1 \cdot d + \ldots + \alpha_n \cdot d^n},
\]

where \( n \) is the dimension of the state vector and the \( \alpha_i \)'s and \( \beta_i \)'s are elements of \( GF(2) \). A realization of an automaton by its transfer function according to Eq. (4) is shown in Fig. 4. Here, the components \( z_1(t+1), \ldots, z_n(t+1) \) of \( z(t+1) \) are determined by

\[
z_n(t+1) = \alpha_n \cdot y(t) + \beta_n \cdot x(t) \quad \text{and} \quad z_i(t+1) = z_{i+1}(t) + \alpha_i \cdot y(t) + \beta_i \cdot x(t)
\]

for \( i = 1, \ldots, n - 1 \) and the output \( y(t) \) is given by

\[
y(t) = z_1(t) + x(t) \cdot \beta_0.
\]
From Eq. (5) we obtain
\[
z_n(t+1) = \alpha_n \cdot y(t) \oplus \beta_n \cdot x(t)
\]
\[
z_{n-1}(t+1) = \alpha_n \cdot y(t-1) \oplus \beta_n \cdot x(t-1)
\]
\[
\oplus \alpha_{n-1} \cdot y(t) \oplus \beta_{n-1} \cdot x(t)
\]
\[
z_{n-2}(t+1) = \alpha_n \cdot y(t-2) \oplus \beta_n \cdot x(t-2)
\]
\[
\oplus \alpha_{n-1} \cdot y(t-1) \oplus \beta_{n-1} \cdot x(t-1)
\]
\[
\oplus \alpha_{n-2} \cdot y(t) \oplus \beta_{n-2} \cdot x(t)
\]
\[
\vdots
\]
\[
z_1(t+1) = \alpha_n \cdot y(t-[n-1]) \oplus \beta_n \cdot x(t-[n-1])
\]
\[
\oplus \alpha_{n-1} \cdot y(t-[n-2]) \oplus \beta_{n-1} \cdot x(t-[n-2])
\]
\[
\oplus \cdots \oplus \alpha_1 \cdot y(t) \oplus \beta_1 \cdot x(t)
\]
(6)
and the state \( z(t+1) = z_1(t+1), \ldots, z_n(t+1) \) of the linear automaton \( L_0 \) with initial state \( z(0) = 0 \) is uniquely determined by the last \( n \) outputs \( y(t), \ldots, y(t-n+1) \) and the last \( n \) inputs \( x(t), \ldots, x(t-n+1) \) of that automaton.

This property will be utilized for correcting an erroneous state of fault-tolerant linear state machines as described in the next section.

III. PROPOSED METHOD

The general principle of the proposed fault-tolerant design for linear state machines with initial state \( z(0) = 0 \) is to implement the output \( y(t) \) as a fault-tolerant signal \( y^{\text{cor}}(t) \) and to use this fault-tolerant signal for state correction. Two modifications are described in this section.

A. Triplication with State Correction

In this subsection fault tolerance for linear state machines by triplication with state correction is described.

In Fig. 5 a linear state machine \( L_0 \) with initial state \( z(0) = 0 \) is triplicated into \( L^1_0 \), \( L^2_0 \) and \( L^3_0 \) and their corresponding output signals \( y^1(t), y^2(t) \) and \( y^3(t) \) are connected to a three-input voter \( V \), which determines the corrected output signal \( y^{\text{cor}}(t) \).

We assume that the state vector has \( n \) components with \( n \geq 1 \).

The corrected output signal \( y^{\text{cor}}(t) \) replaces the three respective output signals \( y^1(t), y^2(t) \) and \( y^3(t) \) of the triplicated linear state machines \( L^1_0 \), \( L^2_0 \) and \( L^3_0 \). Thus, the corrected output signal is used along with \( x(t) \) to determine the corresponding next states of the linear state machines \( L^1_0, L^2_0 \) and \( L^3_0 \).

It was already described in subsection II-B that the state vector of a linear state machine is uniquely determined by the last \( n \) inputs and the last \( n \) outputs. If only one of the linear state machines is (temporarily) erroneous, the output of the voter \( V \) is always correct and therefore we have:

We assume that the state vector of \( L_0 \) is \( n \)-dimensional. If a state vector of only one of the linear state machines is (temporarily) erroneous, this state will be corrected within \( n \) clock cycles, because the components of this state vector depend only on the last \( n \) inputs \( x(t), x(t-1), \ldots, x(t-n+1) \) and the last \( n \) corrected outputs \( y^{\text{cor}}(t), y^{\text{cor}}(t-1), \ldots \) of that automaton.

B. Duplication, Parity Checking and State Correction

In this subsection fault tolerance for linear state machines by duplication with parity checking and state correction is described.

As a general approach fault tolerance by duplication with concurrent checking was introduced in [17], where the considered system \( S \) is duplicated into two identical systems \( S_1 \) and \( S_2 \) which are connected to a multiplexer \( MUX \). Concurrent checking is added either to both the duplicated systems \( S_1 \) and \( S_2 \) or only to \( S_1 \). The output of \( S_1 \) is used as the output of the overall system. However, if an error in \( S_1 \) is detected, the output of \( S_2 \) is used instead. If the considered system is a sequential circuit, the error may be preserved in \( S_1 \) for a long time and after the occurrence of the first fault in \( S_1 \) the overall system is only error-detecting. Now if an error in the second system \( S_2 \) is detected, the system indicates an uncorrectable error and has to stop its operation.

We utilize the shift register structure of linear state machines with initial state \( z(0) = 0 \) to correct an erroneous state during normal operation, and the duplicated system can continue to operate without interruption.

As in the general approach, we duplicate the linear state machine \( L_0 \) with \( n \)-dimensional state vector into two identical copies \( L^1_0 \) and \( L^2_0 \) and add either to both of them or only to the first linear state machine \( L^1_0 \) concurrent error detection by a parity bit \( P^1(t) \) (and \( P^2(t) \)), which, because of the shift register structure of the linear state machine, can be implemented very simple.
We first describe the case where both the linear state machines $L_0^1$ and $L_0^2$ are concurrently checked by parity prediction as illustrated in Fig. 6.

To the first linear state machine $L_0^1$ a first multiplexer $MUX_y^1$ (1), a second multiplexer $MUX_y^2$ (2) and a first 1-bit parity register $P^1$ for storing a first parity bit $P^1(t)$ are added. Similarly, to the second linear state machine $L_0^2$ a third multiplexer $MUX_y^3$ (3), a fourth multiplexer $MUX_y^4$ (4) and a second 1-bit parity register $P^2$ for storing a second parity bit $P^2(t)$ are added. The outputs $y^1(t)$ and $y^2(t)$ of $L_0^1$ and $L_0^2$ are connected to the multiplexers $MUX_y^1$ (1) and $MUX_y^3$ (3). The output of $MUX_y^1$ (1) is the corrected circuit output $y^{cor}(t)$. The output $y^{cor}(t)$ is also input to $L_0^1$ and used for the determination of the first parity bit $P^1(t)$ which is stored in the first parity register $P^1$. The output of $MUX_y^3$ (3) is input to $L_0^2$ and it is also used for the determination of the second parity bit $P^2(t)$ which is stored in the second parity register $P^2$. The output of the multiplexer $MUX_y^2$ (2) is connected to the parity register $P^1$ and via the multiplexer $MUX_y^4$ (4) with the input of the parity register $P^2$. Analogously, the output of the multiplexer $MUX_y^4$ (4) is connected to the parity register $P^2$ and via the multiplexer $MUX_y^2$ (2) with the input of the parity register $P^1$. The value $p_1(t)$ ($p_2(t)$) is the parity of the components of the state vector of $L_0^1 (L_0^2)$ which is determined by an XOR-tree.

If there is no error, we have $y^1(t) = y^2(t) = y^{cor}(t)$ and the control signals of all the multiplexers are equal to 0. The first parity bit $P^1(t)$ is determined from $x(t)$ and from the output $y^1(t)$ of $L_0^1$ and the second parity bit $P^2(t)$ is determined also from $x(t)$ and from the output $y^2(t)$ of $L_0^2$.

Let us assume now that a transient single bit error occurred in one of the flip-flops of $L_0^1$. Then we have $P^1(t) + p^1(t) = 1$ as long as this single bit error is not shifted out of the shift register of $L_0^1$ (see Fig. 4).

Then the correct values $y^{cor}(t) = y^2(t)$ is output by $MUX_y^4$. Also the correct value $P^2(t)$ is output by the multiplexers $MUX_y^3$ and $MUX_y^2$ and stored in the parity registers $P^1$ and $P^2$.

Since the correct output value $y^{cor}(t) = y^2(t)$ is fed back into both the linear state machines $L_0^1$ and $L_0^2$ as long as an error is indicated by $P^1(t) + p^1(t) = 1$, the considered single bit error is shifted out of the shift register of $L_0^1$ (see Fig. 4).

Similarly, it can be shown that all single bit transient errors within $L_0^2$, $P^1$, $P^2$ and the XOR-gates are also corrected at latest after $n$ clock cycles.

Let us consider now that only the first linear state machine $L_0^1$ is implemented with error detection as shown in Fig. 7. If an error is detected within the state of $L_0^1$ or the parity bit of $L_0^1$, the multiplexers $MUX_y$ (1) and $MUX_P$ (2) switch and output $y^2(t)$ and the parity bit $P^2(t)$ as long as the error detection circuitry outputs an error signal, that is, until the error in $L_0^1$ disappears.

A transient error within the state of $L_0^2$ is automatically corrected at latest after $n$ clock cycles since the correct output of the multiplexer $MUX_y$ is fed back to both $L_0^1$ and $L_0^2$.

The parity bit of a linear state machine $L_0$ is determined as

$$P(t) = z_n(t) \oplus z_{n-1}(t) \oplus \cdots \oplus z_2(t) \oplus z_1(t)$$

$$P(t + 1) = z_n(t + 1) \oplus z_{n-1}(t + 1) \oplus \cdots \oplus z_2(t + 1) \oplus z_1(t + 1)$$

and with Eq. (5) we conclude

$$P(t) = P(t) + \Delta P$$

$$= z_n(t) \oplus y(t) \cdot \alpha \oplus x(t) \cdot \beta,$$

where $\alpha = \alpha_1 + \cdots + \alpha_n$ and $\beta = \beta_1 + \cdots + \beta_n$ and

$$P(t + 1) = P(t) + \Delta P$$

$$= P(t) + y(t) \cdot \{1 + \alpha\} \oplus x(t) \cdot \{\beta_0 + \beta\}.$$}

Fig. 6 shows, as already pointed out, an implementation with error detection for both $L_0^1$ and $L_0^2$ and Fig. 7 presents a circuitry where error detection is only implemented for $L_0^1$.

IV. EXPERIMENTAL RESULTS

Since in all proposed designs single transient errors in the state components and in the combinational circuit parts are corrected (at least after $n$ clock cycles), they are compared with known solutions, which also correct state errors and errors in the combinational parts of the circuit.

Linear state machines of different dimensions (from 8 to 256 bits) were chosen and the area overhead of the proposed designs was determined.

The following designs were implemented:

1) The linear state machine is triplicated and the outputs are voted by a single voter. The corrected output $y^{cor}(t)$ is besides $x(t)$ the input of all the triplicated linear state machines.

2) The linear state machine is triplicated and the outputs are voted by three voters. The corrected outputs of the three voters are besides $x(t)$ the input of the corresponding triplicated linear state machines.

3) The linear state machines are duplicated. For error detection two parity bits are implemented.

4) The linear state machines are duplicated. A single parity bit is used for error detection.

The proposed designs are compared with the following well-known fault-tolerant solutions with state error correction:
schemes

Experimental results have shown that the lowest area overhead can be obtained if the linear state machine is duplicated and a single parity bit is used to distinguish which of the duplicated machines is correct.

In this case, an area overhead of 177 % for an 8-bit state vector and 160 % for a 256-bit state vector is achieved.

ACKNOWLEDGMENT

The authors would like to thank V. Petrovic and M. Krstic from IHP Frankfurt/Oder for their helpful suggestions for the synthesis and the reviewers for their comments.

REFERENCES