A Comparison of Fault-Tolerant State Machine Architectures for Space-Borne Electronics

Shailesh Niranjan  
Cirrus Logic, Fremont  
James F. Frenzel, Member IEEE  
University of Idaho, Moscow

Key Words — Fault-tolerant VLSI, radiation-immunity, space electronics.

Summary & Conclusions — Very large scale integrated (VLSI) circuits used in the space & nuclear industry are continuously subjected to ion radiation. As the limits of VLSI technology are pushed towards sub-micron levels in order to achieve higher levels of integration, devices become more vulnerable to radiation induced errors. These radiation induced errors can lead to system failure, particularly if they affect the memory portion of vital subsystems, such as state machine controllers.

This paper explores the use of classical fault-tolerant state machine architectures based on hardware & information redundancy to design radiation-immune controllers. Those architectures particularly suitable for VLSI-implementation using ordinary, low power CMOS technology are identified, with the primary objective of correcting single flip-flop errors. Each architecture was implemented on a set of benchmark sequential circuits and evaluated in terms of circuit-size and maximum path-delay. The best overall architectures, ‘SEU-I TMR’ and ‘Modified Explicit EC’, used a non-redundant excitation circuit and redundant flip-flops, followed by error correction circuitry to tolerate single flip-flop errors.

1. INTRODUCTION

Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEC</td>
<td>explicit error correction</td>
</tr>
<tr>
<td>IC</td>
<td>integrated circuit</td>
</tr>
<tr>
<td>IEC</td>
<td>implicit error correction</td>
</tr>
<tr>
<td>VLSI</td>
<td>very large-scale integrated (circuit)</td>
</tr>
<tr>
<td>SEU</td>
<td>single event upset</td>
</tr>
<tr>
<td>TMR</td>
<td>triple modular redundancy</td>
</tr>
<tr>
<td>MCNC</td>
<td>Microelectronics Center of North Carolina</td>
</tr>
</tbody>
</table>

The advancements in VLSI technology toward lower supply voltages and smaller feature size make today’s circuits more susceptible to errors caused by external factors. These errors, either transient or permanent, can lead to system failure. This is of particular concern for sequential-state machines, which act as controllers in most VLSI systems, because a transient error in a sequential machine has the potential to become permanent if it affects the storage element of the machine. This type of error is very common in VLSI circuits used for space-borne applications, where random high energy cosmic ion strikes cause soft errors in memory elements, referred to as SEU [1]. While there are ways of preventing SEU-induced errors, they involve either costly processes with larger feature sizes or storage elements with higher static power dissipation [2]. This paper concentrates on design techniques for synchronous state machines which tolerate single errors in the state variables. This restriction is quite reasonable, as the effect of an ion particle strike is confined to a region 2 μm in diameter [3]. The use of fault-tolerant architectures to tolerate an SEU-event allows the manufacture of VLSI circuits for mission-critical space applications using conventional, state of the art, low power CMOS processes.

Research on the design of fault-tolerant controllers started in the early 1960’s. Many schemes have since been proposed for fault-masking in sequential machines [4 - 12]. With the development of signature analysis in the 1970’s and the increased importance of testing and testable designs, focus shifted away from fault-tolerant state machines and led to other areas of research. One such area was control-flow checking for synchronous state machines [13 - 15] and is closely related to fault-tolerant design techniques. However, this work is not suitable for achieving on-line fault-tolerance in state machines due to the reliance on off-line techniques for detecting errors. The availability of CAD tools and advances in VLSI technology have rekindled research interest in the classical fault-tolerant design methods [16]. This paper compares the tradeoffs associated with various fault-tolerant state machine designs implemented in standard cell, CMOS technology using the Octtools IC design suite from UC-Berkeley.

Section 2 presents design methods to tolerate single errors in state machines based on the classical fault-tolerant design principles. Two of these methods are based on hardware redundancy principles and the remainder on information redundancy techniques. State machines from the MCNC logic synthesis benchmark suite [17] were implemented using these techniques and the results are presented in section 3.

2. FAULT-TOLERANT DESIGN TECHNIQUES

A finite state machine M is: A 5-tuple \((S, I, O, \delta, \lambda)\).

Notation

- \(S\): a finite, non-empty set of states
- \(I, O\): finite, non-empty set of [inputs, outputs]
- \(\delta: S \times I \rightarrow S\): the next state transition function
- \(\lambda: S \times I \rightarrow O\): the output forming logic function

The symbolic states in \(S\) are encoded by a group of binary state variables. In this paper, the logic block which implements the
Next state transition function is the excitation circuit. The next-state-logic plus the output-forming-block represent the combinatorial logic of the state machine, while the flip-flops which store the state information form the memory section. This paper focuses on methods for tolerating single errors in the state variables, such as might be induced by a SEU. In some cases, the architecture also tolerates certain gate failures in the excitation circuit.

Fault-tolerant designs use additional resources beyond that needed for usual operation. These additional resources can be in the form of hardware, information, or time. SEU-immune state machines can be designed using one or a combination of these additional resources. However, time redundancy is not considered because we are interested in producing state machines which operate at speeds comparable to the non-redundant implementations.

2.1 TMR Architecture

TMR is the most common form of hardware redundancy. In this implementation, the excitation circuit and the state flip-flops are triplicated and the 3 copies of the present state variables are voted on to produce the correct present state. Each module is a complete, non-redundant state machine without the output forming logic. The present state output from the voter is fed back to the 3 modules of the excitation circuit. This design masks any single state variable error caused by a gate failure in the excitation circuitry or the memory portion of the state machine.

Since the emphasis of this paper is on the design of an SEU-immune state machine there is no need to triplicate the excitation circuit. A simpler architecture can be constructed in which the next state is generated from a single, non-redundant, excitation circuit and fed to 3 copies of the state flip-flops. As in the full TMR architecture, the outputs of the flip-flops are voted on to produce the correct present state. This configuration is SEU-I TMR to distinguish it from the full TMR implementation.

2.2 Duplex Architecture

A single fault-tolerant state machine can be designed using 2 copies of fault detecting state machines. Architectures which implement this scheme have been suggested [10, 11]. To perform 1-bit error detection, the Hamming distance between any pair of states must be ≥ 2. This can be done by designing the excitation circuit of each machine to generate a parity bit along with the original state variables, increasing its size compared to a non-redundant circuit.

The next state generated by the excitation circuit is fed into two s-independent sets of state flip-flops. Parity of the state variables is regenerated by 2 copies of error detecting circuits to form error signals, $E_M$ & $E_S$. If both of the error signals are low, indicating a condition of no error, the present state generated from module 1 is routed through the selector circuit to the feedback path of the state machine. In the event that a single error occurs in one of the modules, the corresponding error signal is asserted and the present state from the other module routed to the feedback path. If both $E_M$ & $E_S$ are asserted, indicating faults in both modules, the system can be forced into a fail state by the selector. Faults in the excitation logic block, error detection circuit, and the selector circuits are not tolerated in this architecture.

2.3 EEC Architecture

The EEC architecture was the earliest attempt to apply Hamming error correcting codes to the design of fault-tolerant state machines [4, 7]. The states of the given sequential machine are encoded with Hamming distance 3 codes, providing for 1-bit error correction. The excitation circuit is then designed; due to many don’t-care entries introduced by the redundant state variables, this logic can be implemented efficiently.

The EEC circuit is designed based on the syndrome generation principle. The parity bits are regenerated from the output of the state flip-flops. A bit-wise exclusive-or of the generated parity and the output from the state flip-flops which hold the original parity bits results in a syndrome. The value of the syndrome indicates the state variable in error and is used to correct the original state variables or the parity state variables. The output of the error correction circuit is the correct present state, which contains both the non-redundant state variables and the appended parity.

2.4 Modified-EEC Architecture

In EEC architecture, all of the state variables, including the parity bits, are generated by the excitation circuit using only the outputs from those flip-flops which hold the non-redundant state information. This redundancy in the excitation circuit can be eliminated by using a separate parity-appender circuit to generate the parity state variables. A minimum number of parity bits, as given by the Hamming bound, are appended to the output of a non-redundant excitation circuit to maintain a distance of 3 between any pair of next states.

As with EEC, separate error correcting logic is placed in between the state flip-flops and the excitation circuitry. However, this circuit is simpler than EEC because only the non-redundant variables are corrected; errors in the parity state variables are not corrected because they are not used to generate the next state of the machine.

2.5 IEC Architecture

IEC is also based on Hamming encoding of the states and is very similar to the ideas of Russo [5] and Meyer [12]. Rather than use separate circuitry to correct errors in the state variables, the error correction function is built into the excitation logic of the sequential machine. As a result, adjacent entries in the next state map, corresponding to single state variable errors, must be changed from don’t-cares to the state encoding of the correct next state. For example, if state $A$ is encoded as $<00000>$, then the next state map has 6 entries filled with the correct next state for present state $A$ under each input condition: one entry corresponding to no state variable errors and 5 for each single error. Consequently, the number of don’t-care states is greatly reduced, resulting in a larger excitation circuit.
The advantage of this architecture is that if gates in the excitation circuit are not shared between state variables, then faults introduced by any single gate failure are also tolerated. A corrected version of the present state variables is not available for the output logic. Consequently, the output logic must be designed to use present inputs and present next state values to generate the proper output.

2.6 Qualitative Analysis

A true single fault-tolerant state machine is: one which tolerates any single gate failure in the combinational or memory section of the machine, including the input decoding circuitry. The area overhead associated with such designs can be very large and often unacceptable. More often, fault-tolerant designs cater to a small set of error conditions; in this work, the architectures presented tolerate single, soft or hard errors in state flip-flops. The remaining circuitry is assumed to be fault free.

The full TMR architecture is almost a true single fault-tolerant state machine since any gate failure in the excitation circuit or state flip-flop error is tolerated. Intuitively, this architecture results in an area increase of more than 200% compared to a non-redundant machine. In contrast, the SEU-I TMR architecture has a smaller area overhead since there is a single copy of the excitation circuit. However, the architecture tolerates only a restricted set of faults compared to the full TMR implementation. The basic advantage of the TMR based schemes is the ease and simplicity of the design. The duplex architecture has fewer redundant flip-flops compared to the TMR architecture. However, in this case the error detection and selector circuit are complex. The EEC & IEC architectures are based on information redundancy techniques. These architectures always result in an equal, if not fewer, number of redundant flip-flops compared to the other schemes.

The architectures in this section were applied to the MCNC logic synthesis benchmark state machines. Section 3 compares each method in terms of area and maximum path-delay.

3. PERFORMANCE EVALUATION & ANALYSIS

The MCNC maintains benchmark circuits for researchers interested in test generation and logic synthesis of combinational & sequential circuits. These circuits were assembled from industrial & academic sources from many countries and are widely used to compare results in logic synthesis and optimization methods. State machines from the MCNC 1991 logic-synthesis benchmark-set were implemented to compare the architectures in section 2.

Two important criteria for comparing VLSI implementations are:

- die area occupied by the circuit,
- speed of operation.

Five benchmark state machines (see table 1) recommended by MCNC [17] were simulated, and standard cell layouts generated using the Octtools tool suite developed by UC-Berkeley. The results of these implementations form the basis for a quantitative comparison of the architectures in terms of area & speed.

3.1 State Machine Implementation using Octtools

A 5-step procedure was used in the design of non-redundant state machines:

- State Assignment. State encoding was done with the primary goal of minimizing the combinational logic of the machine, using the program jedi.
- Excitation Circuitry. A behavioral description file which describes the next state behavior of the state machine was generated from the state table. This file was provided to misII, a multiple level, combinational logic optimization program. misII maps the optimized logic into a net-list of standard cells. The standard cell library used for this work was developed at Mississippi State University for a two metal, scalable CMOS process.
- Flip-Flops. Standard cell, set-reset, D flip-flops were externally interfaced to the unplaced next state logic to realize a complete state machine.
- Simulate. A multilevel, functional logic simulator, musa, was used to verify the state machine design.
- Place and Route. VLSI layouts were generated using an iterative standard cell place and route utility wolfe.

Table 1 shows the results of implementing the non-redundant benchmark machines. Because the standard cell library is scalable, dimensions are given in terms of \( \lambda \), rather than \( \mu \). The worst case delay through the combinational logic was found with a 1 pF load on all internal nodes of the circuit.

<table>
<thead>
<tr>
<th>Number of state variables</th>
<th>Area of excitation circuit</th>
<th>Area of state flip-flops</th>
<th>Optimized area of machine</th>
<th>Dimensions of chip</th>
<th>Max Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State dk14</td>
<td>7 states</td>
<td>8 inputs</td>
<td>203040</td>
<td>616x415</td>
<td>69</td>
</tr>
<tr>
<td>State bbara</td>
<td>10 states</td>
<td>4 inputs</td>
<td>582176</td>
<td>816x776</td>
<td>105</td>
</tr>
<tr>
<td>State dk512</td>
<td>15 states</td>
<td>2 inputs</td>
<td>338136</td>
<td>568x852</td>
<td>81</td>
</tr>
<tr>
<td>State cse</td>
<td>16 states</td>
<td>7 inputs</td>
<td>1188336</td>
<td>880x1331</td>
<td>114</td>
</tr>
<tr>
<td>State don</td>
<td>24 states</td>
<td>4 inputs</td>
<td>2357824</td>
<td>1192x1944</td>
<td>128</td>
</tr>
</tbody>
</table>

The fault-tolerant architectures were synthesized & implemented using Octtools with minor modifications to the design flow. For the TMR architectures, the excitation circuit design & implementation is similar to the non-redundant machine. The
A voter circuit in this architecture was described using gates from the standard cell library, along with the state flip-flops. This file was then interfaced to the excitation circuit to realize a complete behavioral description language. The error correction circuit of the duplex architecture is included along with the state flip-flops as in the TMR architecture. However, in the EEC and modified-EEC methods, the error correction logic was described along with the next state logic using the behavioral description language.

3.2 Area Overhead

Optimized areas for each of the architectures were found after generating the layouts for several trial runs. Table 2 shows the fraction increase in the area of the fault-tolerant implementation with respect to the non-redundant implementation. These fractions include the input forming logic, the excitation circuitry, and the flip-flops, but not the output forming logic. The output forming logic is a block of nonredundant circuitry common to all implementations and would distort the comparisons of area overhead if it were included.

- A non-redundant encoding using Jedi and the required parity bits appended manually.
- The state codes obtained were used in the behavioral description file of the excitation circuit.

The error correction circuit of the duplex architecture is included along with the state flip-flops as in the TMR architecture. However, in the EEC and modified-EEC methods, the error correction logic was described along with the next state logic using the behavioral description language.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Description</th>
<th>7 States</th>
<th>10 States</th>
<th>15 States</th>
<th>16 States</th>
<th>24 States</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR</td>
<td>239, 41</td>
<td>222, 31</td>
<td>213, 35</td>
<td>240, 25</td>
<td>267, 21</td>
<td>236, 31</td>
<td></td>
</tr>
<tr>
<td>SEU-I TMR</td>
<td>82, 41</td>
<td>41, 31</td>
<td>60, 35</td>
<td>23, 25</td>
<td>21, 21</td>
<td>45, 31</td>
<td></td>
</tr>
<tr>
<td>Duplex</td>
<td>150, 130</td>
<td>62, 100</td>
<td>118, 130</td>
<td>58, 113</td>
<td>42, 170</td>
<td>88, 129</td>
<td></td>
</tr>
<tr>
<td>EEC</td>
<td>422, 140</td>
<td>86, 90</td>
<td>183, 85</td>
<td>60, 115</td>
<td>82, 170</td>
<td>167, 120</td>
<td></td>
</tr>
<tr>
<td>Modified-EEC</td>
<td>86, 28</td>
<td>41, 55</td>
<td>41, 55</td>
<td>20, 33</td>
<td>30, 61</td>
<td>44, 46</td>
<td></td>
</tr>
<tr>
<td>IEC</td>
<td>860, 60</td>
<td>370, 5</td>
<td>466, 21</td>
<td>360, 33</td>
<td>[Insufficient Memory for Synthesis]</td>
<td>514, 30</td>
<td></td>
</tr>
</tbody>
</table>

*Insufficient memory for synthesis.

Table 2

Fraction of Area and Delay Overhead

[Body of table gives: fraction-of-area (%), delay-overhead (%)]

dk14, cse, the area penalties associated with more state flip-flops and error correction circuitry are equal since they have the same number of state variables. However, there is considerable variation in the total area overhead due to differences in the excitation circuitry. The amount of this logic increase depends upon the state encoding.

- EEC. The area increase is caused primarily by the logic in the excitation circuit for generating the parity state variables. This method is efficient when the amount of redundancy added due to the Hamming code is a minimum. For example, if there are 6 states in a machine, then 3 parity bits are required, while the same number is required for a machine with 16 states. The area occupied by the error correction circuitry in this architecture is, on average, 50% more than that of the duplex architecture. The overall area penalty, due to the redundant excitation circuit and the error correction logic, is more than the duplex architecture and outweighs the advantage of fewer parity state variables.

- IEC. If the gates in the excitation logic are not shared, then any single gate failure in this circuit is tolerated. But, the area overhead for such a design is much higher than the TMR implementation. Machines dk14 and dk512 implemented using information redundancy methods have area overheads higher than the average; this is due to poor optimization of the excitation circuit. For a non-redundant machine implementation, the states are encoded by Jedi to optimize the logic in the excitation circuit. The optimization property in the state codes is lost when parity state variables are appended in the duplex, EEC, IEC architectures. The amount of inefficiency induced in the excitation circuit due to this factor is variable and is machine specific.

- SEU-I TMR and Modified-EEC. These are the two most area efficient schemes to implement radiation-immune state machines using conventional fault-tolerant methods, due to the absence of any redundancy in the excitation circuit. However, as a result, neither of these architectures tolerates any gate failure in the excitation circuit. The increased area of the flip-flops in SEU-I TMR is compensated by the efficient error correction circuit. The SEU-I TMR architecture is easier to implement, particularly the error correction circuit, compared to the modified-EEC architecture.
3.3 Critical Path Delay

The critical path delay dictates the maximum operating frequency of the state machine. To ensure reliable operation, the data inputs to the state flip-flops must be stable for a setup time, \( t_u \), before they can be clocked. After the flip-flop is clocked, the data propagates to the output after a delay of \( t_f \), the flip-flop propagation delay. The output of the flip-flop then becomes the present state input for the excitation circuit. The next state, \( Y \), is generated after a delay through the excitation logic, \( t_c \). The clock period, \( T \), should not be less than the loop propagation delay plus the setup time:

\[
T \geq t_f^{\text{max}} + t_c^{\text{max}} + t_u^{\text{max}}.
\]

For a given set of flip-flop setup and propagation delay times, the propagation delay through the combinational logic determines the clock period. For the fault-tolerant state machine architectures the delay of the error correction circuitry increases the excitation circuit delay, resulting in a lower clocking frequency. The longest path in the combinational logic was identified and the \( t_c \) calculated for each architecture using \textit{misII}. The fraction increase of \( t_c \) in a fault-tolerant machine compared to a non-redundant implementation is referred to as the delay overhead and is in table 2.

The additional delay introduced by the TMR and SEU-I TMR architectures are equal since they have the same error correction circuit. Similarly, the duplex and EEC architectures have almost the same delay overhead. The error correction circuit of the duplex architecture is very slow since the fault has to be detected and then the proper state variables selected. In the EEC architecture, the delay overhead is more due to the redundant excitation circuit.

For the modified-EEC architecture, the error correction circuit delay is less compared to the EEC architecture. This is because the error correction circuit is designed to correct only the errors in the non-redundant state variables. The SEU-I TMR and modified-EEC architecture are comparable in terms of speed of operation, with the TMR-based scheme having a slight edge.

These experiments show that the SEU-I TMR and modified-EEC architectures are the best candidates for building state machines which are tolerant of SEU. The modified-EEC architecture is always the slower of the two, due to the complexity of the error correction logic compared to the voter circuit. In terms of area overhead, the two architectures are typically quite close. The SEU-I TMR architecture generally requires more state flip-flops than the modified-EEC architecture; however, this is often comparable to the overhead of the parity appender and error correction circuitry.

ACKNOWLEDGMENT

This research was supported by US NASA under Space Engineering Research Center Grant NAGW-3293.