JOURNAL OF ELECTRONIC TESTING: Theory and Applications 21, 429–444, 2005

A Comparative Evaluation of Designs for Reliable Memory Systems

G.C. CARDARILLI
Department of Electronic Engineering University of Rome "Tor Vergata", Rome, Italy
g.cardarilli@ieee.org

F. LOMBARDI AND M. OTTAVI
Department of Electrical and Computer Engineering Northeastern University, Boston, USA
lombardi@ece.neu.edu
mottavi@ece.neu.edu

S. PONTARELLI, M. RE AND A. SALSANO
Department of Electronic Engineering University of Rome "Tor Vergata", Rome, Italy
pontarelli@ing.uniroma2.it
marco.re@ieee.org
salsano@ing.uniroma2.it

Received October 5, 2004; Revised March 8, 2005

Editors: C. Metra and R. Leveugle

Abstract. This paper addresses the design of storage systems for operation under critical environmental conditions. For these applications, these systems should have low latency time in access, high performance in throughput and high storage capabilities; therefore, they must be assembled using highly reliable components, while allowing flexibility in design. Commercial Off The Shelf (COTS) components have often been used. A COTS-based architecture is analyzed in this paper; the proposed architecture uses design-level techniques (such as error detection/correction codes and scrubbing) to make commercially available Dynamic Random Access Memory (DRAM) chips tolerant to faults. This paper provides a complete and novel analysis of engineering alternatives which arise in the design of a highly reliable memory system based on Reed Solomon coding. A comparative analysis of methods for permanent fault detection is provided; moreover using a Markovian characterization, different functional arrangements (based on code and scrubbing frequency) are investigated and evaluated.

Keywords: Markov model, reed solomon codes, fault tolerance, mass memories

1. Introduction

Some applications require memory systems capable of storing large amounts of data with tight requirements on high reliability and data integrity. In general reliability and data integrity are dependent on operational features; therefore, mass storage systems which operate in critical environmental conditions require an accurate evaluation of these requirements at design level. A particularly harsh environment is space; in space, high energy particles together with thermal and mechanical stress cause an increase in the failure rate of electronic devices. Moreover, effects which in the past were only observed in space applications, are also occurring in

very deep submicron VLSI chips. Due to the reduced critical charge the occurrence of single event upsets by high energy particles [28] is possible.

The requirements in performance of a mass storage system are low latency in access time, high throughput and storage; however, design specifications are usually complex that an optimal balance among these requirements is very difficult to achieve. To meet high performance, the use of Commercial Off The Shelf (COTS) components is usually preferred over radiation hardened components (which unfortunately have lower performance features and higher costs). However if COTS are used, then the memory system must be made tolerant to faults and errors which are caused by the severe environmental conditions for applications such as avionics. In this case, system-level design methodologies are usually employed.

The proposed architecture [4] improves commercially available Random Access Memory (RAM) chips by applying design-level techniques which permit permanent error detection and correction through appropriate codes [3] and scrubbing [19, 27].

Error Detection And Correction (EDAC) codes improve both the reliability (i.e. the probability that the system will perform its required function over a specified period of time) of the storage system and the data integrity (i.e. the probability that the data in the system is correctly stored for a specified period of time); in this paper, a class of maximum distance separable EDAC codes known as Reed-Solomon (RS) codes, is considered. RS codes guarantee a high level of data integrity; they are widely used both for transmission media and storage system by providing a high level of flexibility in the choice of appropriate datawords and codeword lengths.

The use of additional elements (spares) improves the reliability of the memory by allowing it to tolerate the occurrence of permanent faults; in this approach, spares are inserted. A delay (ideally very small) is also accounted in the normal operation of the system, while purging faulty/defective/erroneous components. Finally, the scrubbing technique of [19] (which consists of periodically reading the content of the memory and correcting possible errors) can be used for improving data integrity by reducing the accumulation of transient faults (such as SEUs) in the memory. Therefore, the reliability of a storage system using these techniques is closely related to the occurrence of permanent faults (e.g. stuck-at [12]), while data integrity is mainly related to the occurrence of transient faults (e.g. SEU [2, 8]). These faults can modify the value of the data stored in the memory elements, (even if partially due to the occurrence of permanent faults). The occurrence of a permanent fault has a twofold impact on data integrity:

- The fault can cause loss of data stored in the affected memory element.
- The correction capabilities of the EDAC code which is employed at system level, can be degraded, because the code must correct errors due to both types of fault (i.e. permanent and transient).

The objective of this paper is to provide a comparative evaluation of the different parameters involved in the design of fault-tolerant memory systems. This evaluation is focused on both hardware design issues (as related to permanent fault detection) and operational design choices (for detecting and correcting the occurrence of transient faults).

The paper is organized as follows: Section 2 gives a brief review of RS codes, while Section 3 describes the general architecture of the proposed RS coded memory. In Section 4, two different erasure detection techniques are presented; in Section 5, a comparison of the different erasure detection methods is provided. Section 6 illustrates the method which is used to model data integrity in the memory system; an analysis which provides an accurate estimate of the different figures of merit, is also introduced. Section 7 describes the results as applicable to satellite-based systems. Finally, in Section 8 the conclusion of this paper is provided.

2. Background

Reed Solomon (RS) codes are a subset of the BCH codes; moreover RS codes are linear block codes [3]. A Reed-Solomon code is specified as RS(n, k), where n represents the number of symbols of m bits (with $n \leq 2^m - 1$) of a codeword and k represents the number of symbols of the related dataword. The encoding process starts from the k data symbols (of m bits each) and adds the parity symbols to construct an n symbol codeword. Therefore, $n - k$ parity symbols are present. A RS(n, k) code is capable to correct up to $2 \cdot er + re \leq n - k$, where er is the number of erasures and re is the number of random errors. For data transmission, a random error occurs when a symbol of the received codeword differs from the transmitted

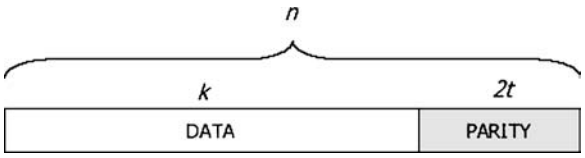


Fig. 1. Reed Solomon Codeword.

codeword at an unknown location. An erasure is said to occur when the channel side information available from the receiver allows to localize the erroneous symbol in the codeword. A Reed-Solomon codeword is shown in Fig. 1: data is left unchanged and parity symbols are appended, hence the Reed Solomon code is also known as a Systematic code.

A widely used Reed-Solomon code is RS(255,223) with 8-bit symbols. Each codeword contains 255 codeword bytes, of which 223 bytes are data and 32 bytes are parity. For this code, $n = 255, k = 223, m = 8, 2t = 32$ and therefore $t = 16$. This code allows to correct any 16 symbol errors in the codeword: i.e. errors (of up to 16 bytes anywhere in the codeword) can be corrected. The ability to encode and decode Reed-Solomon codes is related to the number of parity symbols per codeword. A large value of t means that a large number of errors can be corrected (also requiring more time for the encoding/decoding process). A symbol error occurs when either one bit in a symbol is erroneous, or when all bits in a symbol are erroneous. For example RS(255,223) can correct 16 symbol errors. In the worst case, 16 bit errors may occur, each in a different symbol (byte). Therefore, the decoder will correct 16 bit errors. In the best case, 16 complete byte errors occur, so that the decoder corrects 16×8 bit errors. Therefore Reed-Solomon codes are particularly well suited for correcting burst errors (i.e. a series of bits in the codeword are received in error). Reed-Solomon algebraic decoding procedures can correct errors and erasures. An erasure occurs when the position of an erroneous symbol is known. A decoder can correct up to either t errors, or $2t$ erasures. Erasure information can often be supplied by the demodulator in a digital communication system, i.e. the demodulator "flags" the received symbols that are likely to contain errors. For a memory module, an erasure can be effectively considered as a memory chip failure. When a codeword is decoded, there are three possible outcomes:

1. $2re + er < 2t$ (re errors, er erasures), then the original transmitted code word will always be recovered,
2. The decoder will detect that it cannot recover the original code word and it will flag it out.
3. The decoder will misdecode and recover an incorrect code word with no indication; this event is generally referred to as miscorrection.

The probability of each of the three above events depends on a particular Reed-Solomon code and the number and distribution of errors. Reed-Solomon encoding and decoding can be performed in software or using a special purpose hardware. Reed-Solomon codes are based on Galois or finite fields and a Reed-Solomon codeword is generated using a special polynomial. All valid codewords are exactly divisible by the generator polynomial. The general form of the generator polynomial is given by:

$$g(x) = (x - \alpha^i)(x - \alpha^{i+1}) \ldots (x - \alpha^{i+2t}) \quad (1)$$

and the codeword is constructed as follows:

$$c(x) = g(x) \cdot i(x) \quad (2)$$

where $g(x)$ is the generator polynomial, $i(x)$ is the information block, $c(x)$ is a valid codeword (referred to as a primitive element of the field). The received codeword $r(x)$ consists of the original (transmitted) codeword $c(x)$ with possible errors, i.e.

$$r(x) = c(x) + e(x) \quad (3)$$

A Reed-Solomon decoder will try to identify the position and magnitude of up to t errors (or $2t$ erasures) and correct them. For a memory system, the following assumptions are applicable:

- Transient faults (e.g. SEU) can occur in an unknown location of a codeword, therefore they can be considered as random errors.
- Permanent faults (e.g. stuck-at 0/1) can be located in the memory system by utilizing either self-checking circuits, or on-line testing (refer to Section 4 for more detail); therefore, they can be effectively considered as erasures.

The location of each permanent fault represents a tight requirement for the error correction capabilities of RS codes. While every permanent fault is not localized, the error correction algorithm assumes this error to be random, thus degrading the correction capabilities

of the code. When the permanent fault is located, the capabilities of the RS code to correct an error (which has occurred in a location) can be fully exploited. Permanent fault detection methods are presented in Section 4 together with designs and a comparative evaluation of the different parameters involved in this process. RS codes are suitable for highly reliable memory systems due to their reconfiguration capabilities, as described in Section 3. Finally for random errors, scrubbing can be utilized to take into account accumulation in a codeword [19]. Memory scrubbing consists of periodically reading a codeword, correcting the possible erroneous symbols and rewriting the corrected codeword in the same memory location, thus improving the data integrity of the memory. This technique has three drawbacks:

1. Hardware overhead due to the logic circuitry as required to perform the operation.
2. Increase in the average memory access time with scrubbing frequency.
3. Increase in power consumption with scrubbing frequency.

To avoid the second and third drawbacks requires scrubbing to be accurately evaluated with respect to memory performance. A correct balance in scrubbing frequency with application specifications allows to reduce its negative impact.

The parameters for evaluating data integrity of a memory system based on RS codes can be extracted starting from the architectural issues described previously. Such an evaluation can be performed using a Markovian analysis and related model, as described in Section 7.

3. Architectural Considerations

The design of a storage system made of RAM chips requires an accurate characterization of the COTS components in the environment in which they are expected to operate. The effects of ionizing radiation on memory chips have been extensively analyzed in the literature [6, 29]. Several EDAC-based architectures have been proposed for hardening commercial DRAM memories [7, 9, 10, 16, 18]. Initially introduced in [4], the proposed approach relies on a memory architecture in which EDAC codes are utilized to address issues related to both permanent and transient faults. As shown in [4], error correcting codes increase fault tolerance of a storage system with respect to both reliability and data integrity. The storage system is usually made of the following components:

- a Random Access Memory (RAM) array (made of several COTS chips);
- Control circuitry to interface the memory bank to other components;
- A Reed-Solomon [11,23] coder-decoder which adds redundancy to the data stored in the RAM;
- A Scrubbing Block to perform this operation in the memory system;
- An Erasure Detection Block to detect permanent faults in the RAM chips.

The main hardware overhead introduced by the proposed architecture is related to the Reed Solomon codec. Both FPGA [1, 26] and ASIC [5, 13, 17] implementations of an RS codec can be found in the technical literature. For evaluating and comparing the hardware overhead the ASIC implementation of [5] can be considered; this implementation has a feature size of 0.18 μm with an an area of 1.5 mm^2. A single 144 Mbit DRAM [17] chip (with feature size of 0.13 μm) has an area of 121 mm^2, therefore for a single chip with a small feature size, the overhead is approximately 1%. The overhead due to the codec is even less for a 4.5 GB memory (i.e. 256 chips).

As for the performance degradation introduced by the coding/decoding process , the largest delay is accounted in the decoding operation. This requires filling a pipeline at the first access; therefore, a number of clock cycles (proportional to the number of symbols of a codeword) are required as initial latency. However for burst accesses, the decoding can produce a decoded bit per cycle. FPGA-based implementations require lower clock frequencies (for [26] 110 MHz and for [1] 120 MHz on a single channel). An ASIC implementation can run at 770 MHz [13], achieving a throughput of 6 GBits/s for burst accesses.

The degradation in power consumption due to the proposed architecture is closely related to specific design implementations. At high level, few general considerations are applicable: the power increase is mostly related to the RS decoder; fine tuning of the scrubbing frequency can be implemented for saving power. Moreover, a power reduction in the internal refresh circuitry could be achieved as reported in [10].

From an architectural standpoint, the arrangements of the RAMs permit to implement the Reed Solomon (RS) code with different dataword and codeword

lengths. The selected length depends on the reliability and data integrity requirements (as discussed in more detail next). For a specific RAM scheme, different RS codes can be implemented [4]; for example the RS codes with $n, k = (18, 16)$ $(36, 32)$ $(72, 64)$ $(144, 128)$ have the same k/n ration, i.e. 0.89. The overhead is constant with respect to the selected code and a RAM array consisting of 144 rows can be used to implement all of them. k/n accurately accounts for hardware overhead of the proposed approach because for example, a 4.5 GB memory would have a storage capacity of 4 GB. These characteristics (constant ratio and the use of the same RAM scheme) allow to on-line reconfigure the Reed-Solomon codecs. The RS code reconfiguration [16] can be then used when a permanent failure occurs in a memory. For example, if the memory is initially configured with an RS(18,16) code and a permanent failure occurs, then the code cannot correct any random error. Therefore, the memory should be reconfigured for a new RS code; in this case, a RS(144,128) code can be used to correct 8 erased symbols and up to 4 random symbol errors. If a non reconfigurable RS code is used, then the memory would be permanently faulty; a possible strategy is to substitute the faulty module or chip with a spare. With a dynamically reconfigurable RS codec, if a permanent fault is detected in a codeword, then the content of a faulty memory location (together with the contents of other codewords) can be coded again using a RS code of higher correction capability.

This process requires a buffer to temporarily store the initial codewords for generating a longer codeword after reconfiguration. The use of a higher RS code involves also a higher number of symbols. However, the fixed ratio $\frac{k}{n}$ allows the use of a higher RS code with no symbol overhead.

As an example consider the reconfiguration from RS(36,32) to RS(144,128); assume that starting with a RS(36,32) coding scheme, after some time, three permanent failures (three erasures) are detected. All data stored in the memory module is converted from RS(36,32) to RS(144,128): o or example, four 36-byte codewords are read and decoded and the 128 data bytes are coded into a codeword of 144 bytes.

This procedure allows to preserve the data stored in the memory. However, if the number of erasures is greater than the error correction capability of the code, (for example, 5 erasures for the RS(36,32) code), then the data stored in the codeword is unrecoverable. The functionality of the memory element can be restored using a code with better error correction capabilities. The use of longer codewords improves the integrity of the stored data, but t may possibly degrade performances in terms of latency (as the decoder latency depends on the codeword length).

A scrubbing block (and its relation to the RS codec) is shown in Fig. 2. When a scrubbing operation is performed, the ready signal provided by the scrubbing block is low. Therefore, the external interface is made aware that the memory is ready to meet no user request. At the same time, the enable signal is raised high; so, the RS codec internally switches the address bus. During scrubbing, the address and the control signal are provided to the memory modules (such as the signal from the scrubbing block that during normal operation, the codec receives from the external interface). After generating the read command from the scrubbing block, decoding of the codeword is performed by the codec. After this operation, the scrubbing block generates the write command and the recoded word is rewritten in the memory.

4. Erasure Detection Methods

The last block of the storage system is referred to as the Erasure Detection Block. The design of this block depends on the selection of the permanent fault detection method. In the following subsections, two methods are presented:

1. IDDq-Based Erasure Detection
2. Functional-Based Erasure Detection

4.1. IDDq-Based Erasure Detection

Locating permanent faults can be achieved using different methods. A permanent fault in a memory chip can be detected by monitoring the quiescent supply current (I_{ddq}) [21]. To fully utilize the current sensors for locating faults, the behavior of the memory in the presence of permanent faults and their impact on the quiescent supply current, must be analyzed. Therefore, different types of permanent fault and corresponding parameters must be considered when designing a storage system. For selecting a desirable detection method, it is important to define a fault set and its occurrence probability. The considered fault set is defined as follows:

1. Faults affecting a single memory cell usually cause an anomalous increase of the quiescent supply

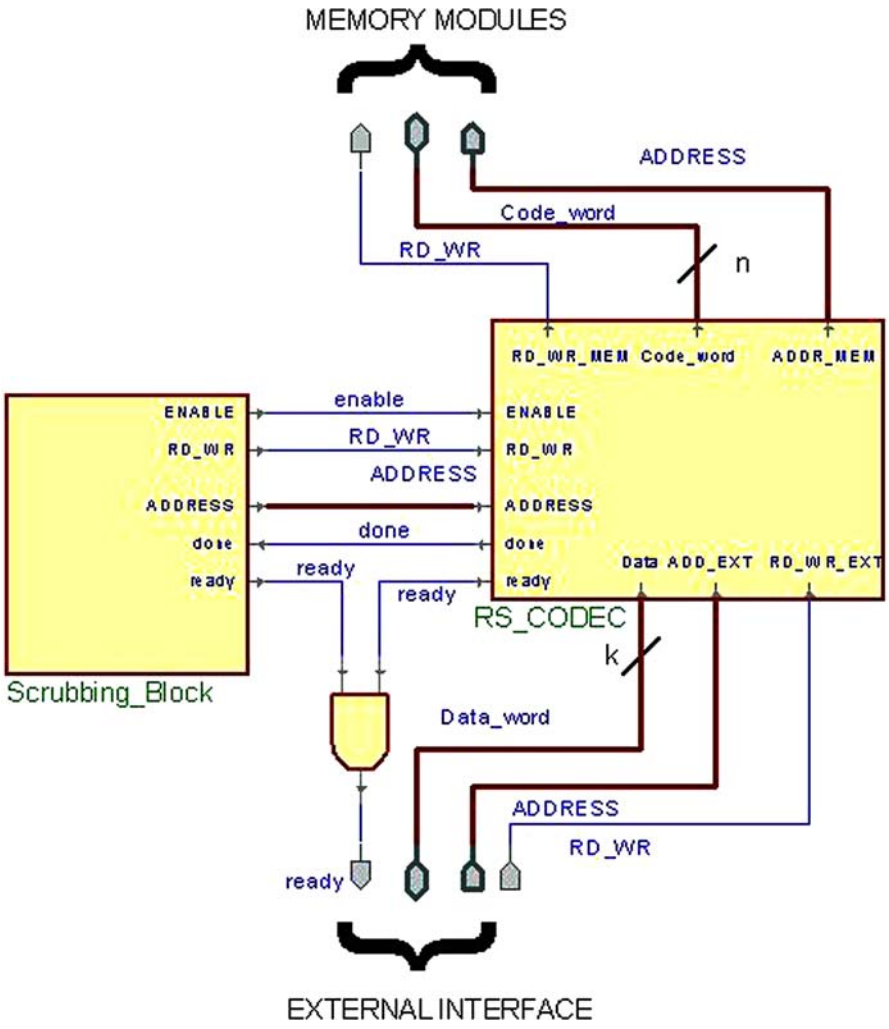


Fig. 2. Scrubbing block.

current, therefore an error location signal can be directly sent to the Reed-Solomon codec for locating erasures. This error location signal is high only when the memory cell affected by the fault is activated, i.e. the faulty cell is located. So, if a fault free memory cell is addressed, then the output of the IDDq tester is low, i.e. the memory location in question is not affected by a permanent fault. When a cell which is affected by a fault is addressed, the quiescent supply current increases, signaling the occurrence of a permanent fault.

2. A subset of single memory cell faults, such as stuck-open faults, (see [14]) can not be detected by IDDq. In this case, the Reed-Solomon codec is still able to correct the error in the codeword, because it is considered like a temporary fault. This type of fault (even if permanent) is corrected; moreover, the rate of occurrence of these faults λ_u (undetected permanent faults) is smaller than λ and in practice, they can be neglected. Otherwise, the Markov chain described in Section 6 must be modified.
3. Faults affecting the address decoding logic (such as row or column decoders) or other types of fault in the control circuitry of the memory chip, can be detected by IDDq testing even if they may affect different memory locations. The computation of λ_e (as used in Section 6) for modeling permanent faults must take into account the occurrence of this type of fault; this is accomplished as follows: as the

evaluation is performed on single codewords, λ_e (as related to faults of type 1) must be added to $k\lambda_a$, where λ_a is the occurrence rate of faults on the decoding logic, and k is the ratio between the number of rows (or columns) affected by the fault and the total number of the rows (columns) of the memory chip.

4.2. *Functional-Based Erasure Detection*

The second method for detecting a permanent fault is based on a functionality test of the memory chip. The fault location procedure starts upon the RS decoder detecting an erroneous codeword. The location procedure consists in correcting the codeword, writing it back to the same location and reading it again; if at completion of this process an error is still present, then the considered location is assumed to be affected by a permanent fault (i.e. erasure). Moreover it is additionally assumed that the probability that during the execution of the described procedure a SEU affects the considered memory location, is negligible. Once a memory address is diagnosed as affected by a permanent fault, the location of the faulty symbol(s) inside of the codeword is found by comparing the corrected codeword with the codeword stored in the memory. The address of the codeword affected by the permanent fault is stored, together with the error location word (which identifies the erased symbols inside the codeword). The block (Fig. 3) reads the address of the requested codeword and the address provides the decoder with the location of the erasures inside the codeword.

This block (Fig. 3) has a Content Addressable Memory (CAM), i.e. the memory provides the data containing the erasure location when the requested address is present in the address match record. The table in Fig. 4 shows the content of the CAM.

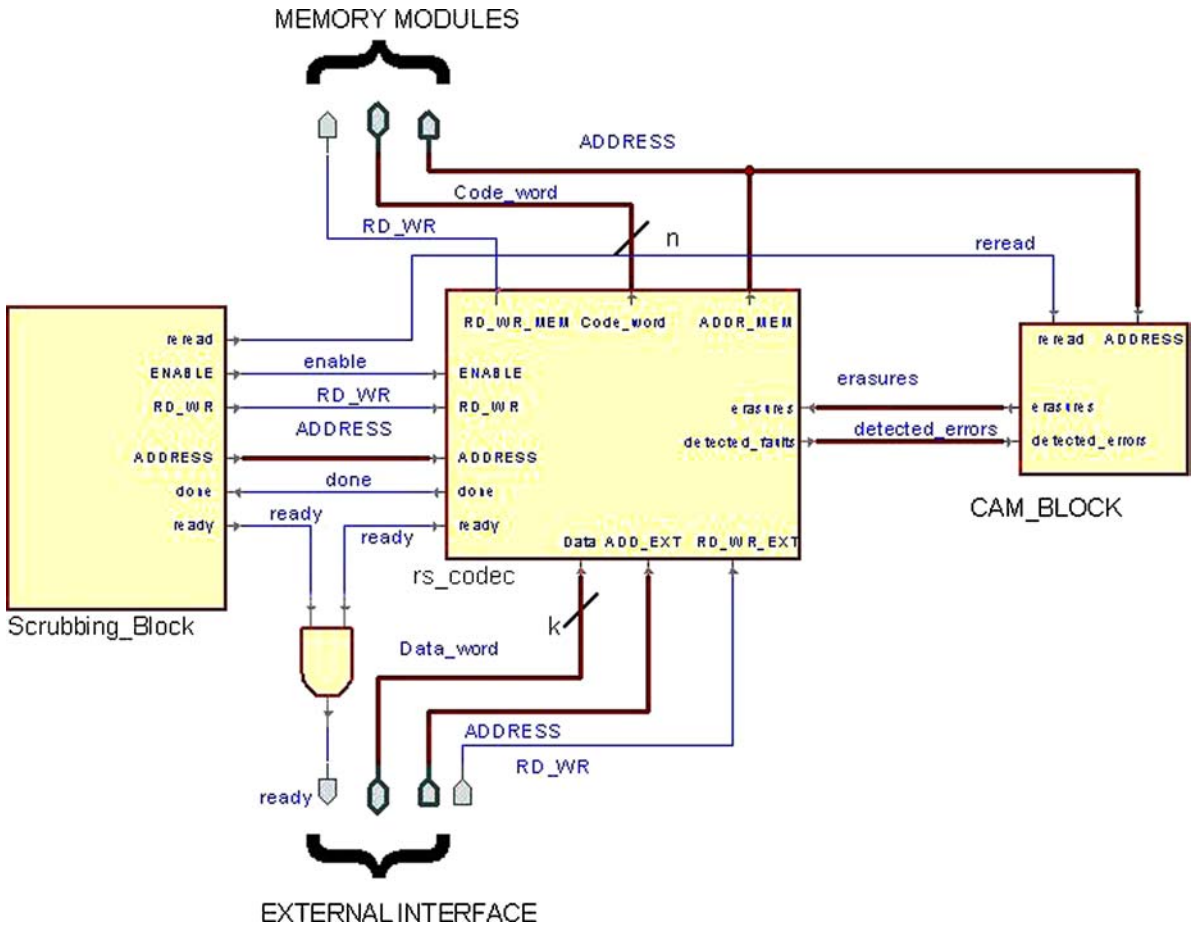


Fig. 3. Functional test block.

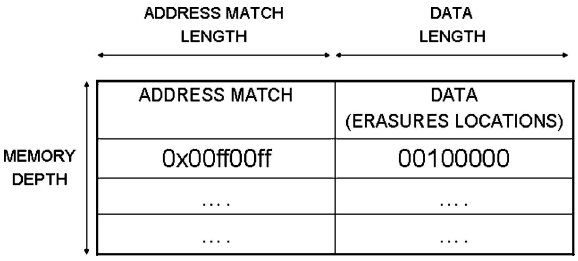


Fig. 4. Table stored in the CAM.

The column on the left hand side of the table represents the addresses in which at least an erasure is detected, while the second record contains the erasure locations. As an example, in the Table of Fig. 4 the address of a codeword (affected by a permanent fault at hexadecimal address 0x00FF00FF) and the corresponding erasure location word are presented. The codeword is composed of eight symbols, the third symbol is affected by an erasure. This method is suitable for the detection of a permanent fault of type (1) and (2) as described in the previous section; however, it is not appropriate for a fault of type (3). While for a fault of type (1) or (2) only a location of the CAM is occupied, the number of entries of the CAM for a fault of type (3) is dependent on the number of memory cells which are affected by this fault.

5. Comparison of Erasure Detection Methods

Performance of the proposed detection methods (with respect to detection capabilities and impact on the overall system) is dependent on the type of permanent fault, and its probability of occurrence. The detection method presented in subsection 4.1 is not useful for a permanent fault of type 2). The probability of occurrence of this type of fault depends on the manufacturing process of the memory chip. An ad-hoc qualification phase (such as proposed in [2, 22]) could be required for the IDDq-based method. These techniques unfortunately require the use of ad-hoc current sensors, which must be correctly tuned to increase their detection capability (for example the threshold at which the sensor must signal the presence of a fault, is different among memory chips). Finally, the current sensors can degrade the performance of a memory, due to their presence in the path between the board power supply and the effective voltage V_{DD} at which the memory chips are driven.

Differently from IDDq-based erasure detection, the method described in Section 4.2 can be applied to different memory chips with no modification to the detection block. This method is less dependent on the electrical characteristics of the memory chip. However, some limitations are encountered when this method is applied. The erasure detection method based on a functional test may result into a large number of possible erasure locations which are dependent on the depth of the Content Addressable Memory used to implement this block. The depth of this memory must be determinated based on the permanent fault occurrence probability. As an example in a storage system of 4 GB protected by a RS(36,32) code (for a 4.5 GB of total memory), there are 2^{32} addressable bytes which correspond to 2^{27} codewords. Therefore, the CAM must have an "address match" field size of 27 bits and a "data" field size of 32 bits. The depth of the CAM corresponds to the maximum allowed number of detectable erasures. Permanent faults of type 3) are very critical, because they require a number of entries in the table stored in the CAM (as related to the number of locations affected by each fault). If the fault occurs in the decoding elements of a memory chip, then this number can be rather large and in the worst case, it can be as large as the entire addressable space of the memory. An approximation to the hardware complexity of the CAM module can be established. The CAM module can be realized using Look-Up Tables (LUTs) FPGAs (such as the Xilinx Virtex II FPGA) configured as SRL16 blocks (for a more detailed discussion of this feature please refer to [25] and [24]). The hardware complexity of this module can be evaluated as number of required LUTs. The number of required LUTs is given by:

$$\#LUT = D \cdot \left(\left\lceil \frac{M}{4} \right\rceil + \left\lceil \frac{n}{16} \right\rceil \right) \tag{4}$$

If a probability less than $P_{\text{full}} = 10^{-9}$ is required for the CAM, then at the end of the mission, the following inequality can be used to establish its depth:

$$P(depth, T_{EOM}) = (1 - e^{C\lambda_e T_{EOM}})^{\text{depth}} < P_{\text{full}} \tag{5}$$

where M is the address match size, i.e., $\log_2(C/k)$, C is the storage capacity of the system (in bytes), D is the depth of the CAM corresponding to the maximum number of detectable erasures, k is the number

of symbols in a dataword, n is the number of symbols in a codeword, T_{EOM} is the End Of Mission time, $P(depth, T_{EOM})$ is the probability to have a number of $depth$ erasures, in a memory of C bytes, at the End Of Mission. However, even if the CAM is full, the RS codec can still correct permanent faults, because they are considered like transient faults. Another drawback of the functional test based method is related to the off-line time of the memory due to scrubbing and test operations. During these operations, the RS codec corrects transient faults and/or detects the permanent faults in the addressed memory location; hence, the user operations of memory Read and Write are blocked. If scrubbing is performed together with functional based erasure detection, the off-line time increases; if either a transient or a permanent fault has occurred in the addressed memory location, the procedure requires a new reading operation. The detection of this type of fault (permanent or transient) is provided only by the second read operation. The off-line time is the sum of the times for the decoding and coding operations (in case of IDDq-based testing), i.e. the time implied for the first decoding, the time for coding, and if an error is detected, the time for the second decoding operation. The probability of performing the write and the second read (recoding and decoding) operations is dependent on the frequency $\lambda + \lambda_e$. The algorithm for functional erasure detection assumes that the pipeline in the operation of the coding and decoding processes can not be easily implemented. Only at the and of the first decoding operation, the algorithm can decide whether another address can be processed, or the write and a second read operations must be performed to detect and locate permanent faults. An architectural solution based on pipelining the operation (together with a flushing procedure) can reduce the possible drawbacks. The RS codec can be implemented by a pipeline architecture, and an additional flushing signal can be used to reset the register in the pipeline (if an erroneous word is detected), i.e. there is a need for performing a write and a second read operations on the same address. This architectural solution can reduce the impact of functional test on the off-line time, but it requires the design of a rather complex erasure detection block as well as the RS codec. In conclusion, the choice of one of the two proposed methods is closely related to the following features:

- The availability of proper electrical characteristics for the memory chip in the presence of faults to tune the IDDq tester. If these parameters are not available, then the use of the functional test approach is much likely imperative.
- A high probability of occurrence of a permanent fault of type 2) discourages the use of an IDDq-based test, while an high occurrence of faults of type 3) increases the depth of the CAM used for the functional approach (therefore affecting the hardware complexity of this block).
- The IDDq-based solution is preferable if the system must have a low off-line time.

A detailed Table 1 summarizing the results of this section is reported: "+" ("−") sign identifies the positive characteristic (negative) of a method.

6. Markov Modeling

A RS coded memory system implementing a scrubbing technique can be described using a Markov model [20]. A Markov model is particularly suitable for establishing and evaluating data integrity as BER at a given time T. The states $S(er, re)$ of the Continuous Time Markov Chain (CTMC) which can be associated with the system temporal evolution, can be uniquely identified by the indices er and re, which represent the number of random errors and erasures in the various

Table 1. Characteristics of the proposed methods.

	Kind (1) fault	Kind (2) fault	Kind (3) fault	HW complexity	Off-line time	Availability of electrical characteristic
IDDq based erasure detection	+	−	+	+	−	− −
Functional based erasure detection	+	+	− −	−	−	+

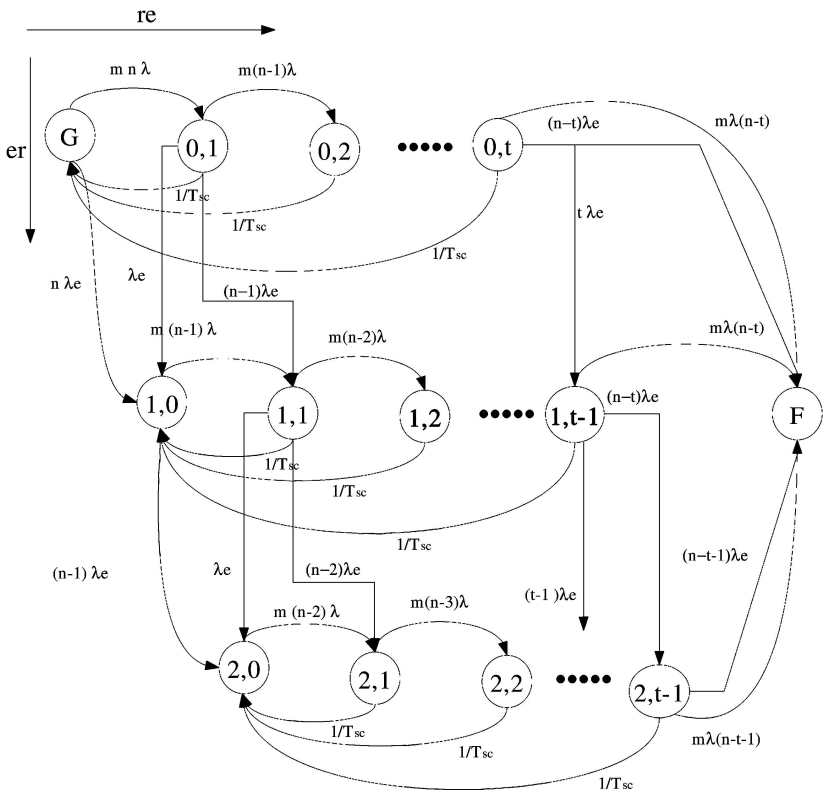


Fig. 5. Markov model of a RS code.

symbols of a codeword at time T. The start state at $T = 0$ (or Good state) is given by $G = S(0, 0)$ in which $er = re = 0$, while the unrecoverable error state (or Fail state) is given by $F = S(er, re)$ (in this state, $2 \cdot er + re > n - k$). A transition between states represents the rate of occurrence of either transient, and/or permanent faults, or a scrubbing operation. If λ denotes the SEU rate affecting a single bit of a symbol, λ_e as the permanent fault rate per symbol and T_{sc} as the scrubbing operation period, then the following transition rates can be defined:

- $r_{re}(er, re) = m \cdot \lambda \cdot (n - er - re)$ is the transition rate from state $S(er, re)$ with er erasures and re random errors to state $S(er, re + 1)$ with er erasures and $re + 1$ random errors.
- $r_{er}(er, re) = \lambda_e \cdot (n - er - re)$ is the transition rate from state $S(er, re)$ with er erasures and re random errors to state $S(er + 1, re)$ with $er + 1$ erasures and re random errors.
- $r_{er*}(re) = \lambda_e \cdot (re)$ is the transition rate from state $S(er, re)$ with er erasures and re random errors to state $S(er + 1, re - 1)$ with $er + 1$ erasures and $re - 1$ random errors. In this case a permanent fault affects a symbol previously affected by a random error.
- $r_{sc} = \frac{1}{T_{sc}}$ is the transition rate from state $S(er, re)$ with er erasures and re random errors to state $S(er, 0)$ with er erasures and 0 random errors. In this case, the scrubbing operation can correctly rewrite all symbols in the codeword affected by random errors, but it cannot rewrite the correct values of symbols affected by erasures (as erasures represent permanent faults).

In the previous definitions, it has been assumed that the probability that a bit flip affects an already affected symbol is negligible, and it can be omitted. Starting from these assumptions and the definitions of rates, a Markov chain as shown in Fig. 5 can be obtained.

In Fig. 5 an RS(n,k) code is used to correct up to t random errors over a fixed T_{SC} period. Due to the correction capabilities, as long as the code is correctable the number of erroneous bits is 0. When the codeword is not correctable, the number of erroneous bits can

be roughly assumed to be equal to the number of bits making the $n-k$ symbols (i.e. the Hamming distance between the two codewords). Thus, the BER of the memory system can be defined as the probability of the codeword of being not correctable, i.e. the product of the probability of being in the F state of the Markov chain (denoted by $P(F)$) and the number of bits in the Hamming distance with the closest symbol. Therefore:

$$BER = m \cdot \frac{(n-k)}{k} \cdot P(F) \tag{6}$$

$P(F)$ can be obtained from the solution of the Markov model. An n-state Markov model leads to a set of n-coupled differential equations. These equations can be represented with vector notation. If the states $S(er, re)$ are ordered from 0 (the G state) to n (the F state), and on the assumption that the vector $P(t) = [P_{S(0)}(t), P_{S(1)}(t), \ldots, P_{S(n)}(t)]$, where $P_{S(i)}(t)$ is the probability of being in state $S(i)$ at time t, then the set of differential equations is given by $P'(t) = \mathbf{A}P(t)$. The matrix $\mathbf{A}$ consists of the transition rates given above. In particular, a generic element $a_{i,j}$ with $i \neq j$ represents the transition rate from state i to state j, while the element $a_{i,i}$ represents the rate related to the probability of permanence in the i-th state, i.e. it can be defined as $a_{i,i} = -\sum_{j \neq i} a_{i,j}$.

$$\begin{pmatrix}
a_{1,1} & 0 & 0 & 0 & 0 & \frac{1}{T_{sc}} & 0 & 0 & \frac{1}{T_{sc}} & 0 \\
n\lambda_e & a_{2,2} & 0 & 0 & 0 & \lambda_e & \frac{1}{T_{sc}} & 0 & 0 & 0 \\
0 & (n-1)\lambda_e & a_{3,3} & 0 & 0 & 0 & \lambda_e & \frac{1}{T_{sc}} & 0 & 0 \\
0 & 0 & (n-2)\lambda_e & a_{4,4} & 0 & 0 & 0 & \lambda_e & 0 & 0 \\
0 & 0 & 0 & (n-3)\lambda_e & a_{5,5} & 0 & 0 & 0 & 0 & 0 \\
nm\lambda & 0 & 0 & 0 & 0 & a_{6,6} & 0 & 0 & 0 & 0 \\
0 & (n-1)m\lambda & 0 & 0 & 0 & (n-1)\lambda_e & a_{7,7} & 0 & 2\lambda_e & 0 \\
0 & 0 & (n-2)m\lambda & 0 & 0 & 0 & (n-2)\lambda_e & a_{8,8} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & (n-1)m\lambda & 0 & 0 & a_{9,9} & 0 \\
0 & 0 & 0 & (n-3)m\lambda & (n-4)(\lambda_e + m\lambda) & 0 & (n-2)m\lambda & (n-3)(\lambda_e + m\lambda) & (n-2)(\lambda_e + m\lambda) & 0
\end{pmatrix}$$

where $a_{i,i} = -\sum_{j \neq i} a_{i,j}$.

Three cases are considered and studied in more detail:

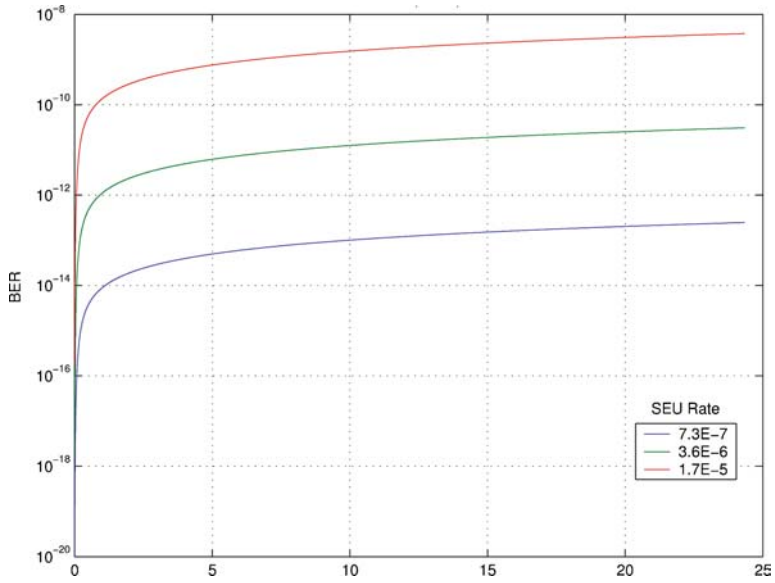
1. Comparison between RS(18,16) and RS(36,32) for BER(t) with no scrubbing and variable SEU rate,
2. Comparison between RS(18,16) and RS(36,32) for BER(t) during solar flares under different T_{sc} periods,
3. Analysis of RS(36,32) for BER(t) by considering the occurrence of permanent faults.

7. BER Analysis and Evaluation

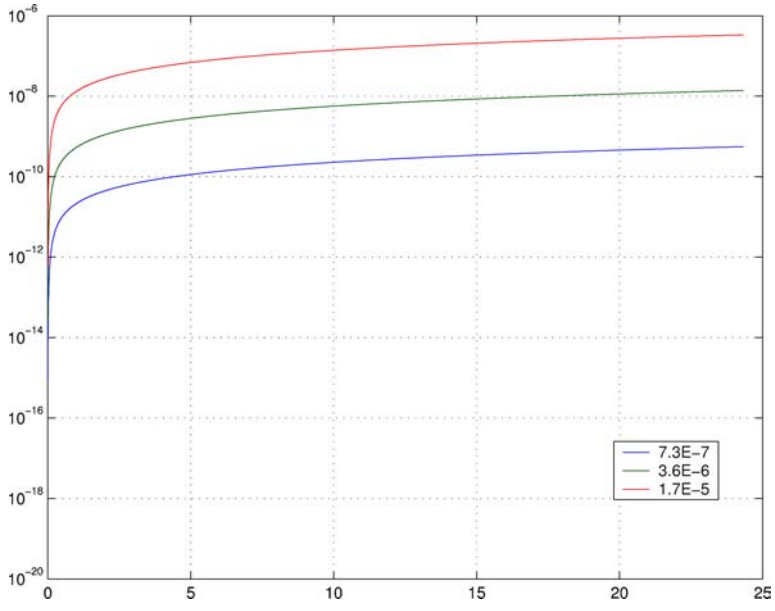
In this section, the evaluation of the BER of a storage system with RS codes is reported. The evaluation is performed using a numerical solution of the set of differential equations and computing the BER as per Eq. (6). The evaluation is pursued by considering different scenarios as follows:

1. The physical environment in which the system operates, i.e. the permanent and transient failure rates,
2. The choice for the RS code, i.e. the data-word and code-word lengths,
3. The periods for the storage time, the scrubbing frequency and the mission time
4. System performance (latency, storage capabilities, mission specs)

As for the transient fault rate, a space environment is assumed throughout as an example of an harsh environment in which high reliable memory systems are often required to operate. In an interplanetary space, a background rate of $7.3 \cdot 10^{-7}$ errors/bit/day is usually considered; this can occasionally increase up to $1.7 \cdot 10^{-5}$ errors/bit/day during solar flares. The rate of permanent faults depends on the reliability of the memory chips and can be evaluated using the models of [4, 15]. The first step in the evaluation process is performed by computing the element $a_{i,i}$ of the matrix $\mathbf{A}$, starting from the transition rates r_{re}, r_{er} r_{er*} r_{sc} described in the previous section. As an example for the RS(36,32) code, the matrix $\mathbf{A}$ is reported below:



(a)



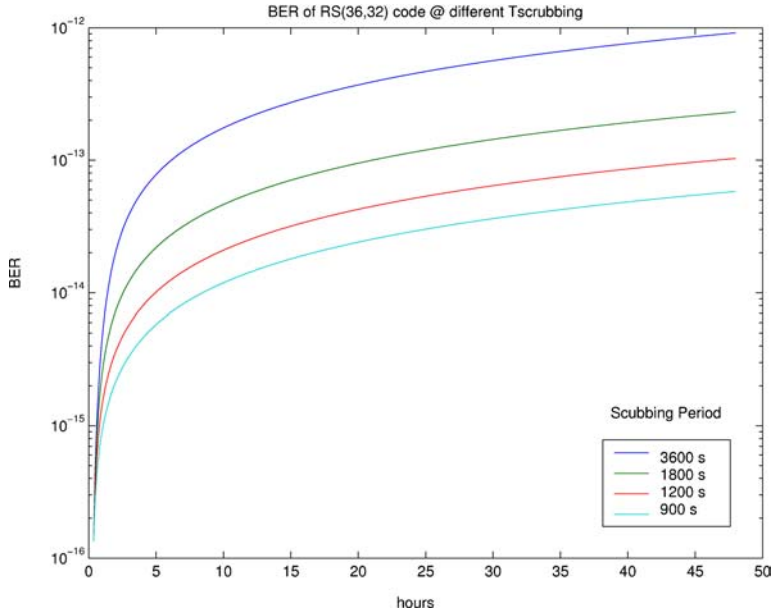
(b)

Fig. 6. BER of RS(36,32) and RS(18,16) codes.

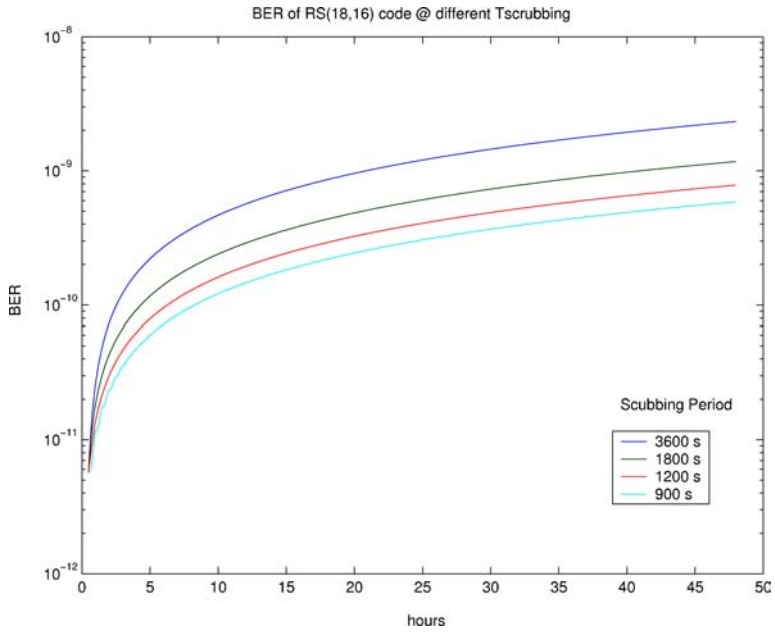
In the evaluation of cases (1) and (2) above, data is assumed to be stored in memory for two days ($T_{st} = 48h$) and, therefore an estimate for the BER during this interval is pursued. These are useful examples to show the flexibility of the proposed method when different design parameters are considered. The first evaluation has been performed on the RS codes RS(18,16) and RS(36,32) with no scrubbing, no permanent fault and with a rate of transient faults given by $\lambda \in [7.3 \cdot 10^{-7}, 1.7 \cdot 10^{-5}]$. Fig. 6 shows the BER of these two codes.

The following considerations can be drawn using the obtained data:

1. At a small SEU rate, the RS(36,32) code provides a BER less than 10^{-12} with no scrubbing.



(a)



(b)

Fig. 7. BER of RS(36,32) and RS(18,16) codes with different T_{sc}.

2. The use of the RS(18,16) code with no scrubbing should be avoided, because its BER is up to 10^{-10} even with a very small SEU rate.
3. The use of the RS(36,32) code with no scrubbing can be considered provided it is assumed that during the mission of the satellite, the frequency of the solar flares is negligible.

Fig. 7 focuses on the behavior of the RS(36,32) and RS(18,16) codes during solar flares. The parameter λ

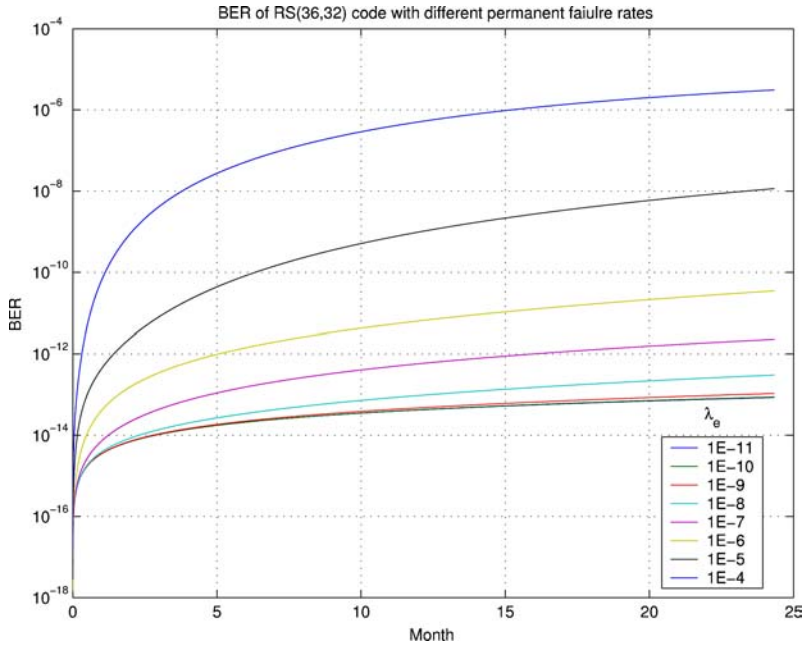


Fig. 8. BER of RS(36,32) code with different λ_e.

is fixed to $1.7 \cdot 10^{-5}$, this is a very large SEU rate. The BER with different T_{sc} is evaluated to tune the scrubbing frequency with respect to the space environment. The results of the BER evaluation for the two codes are reported under different scrubbing frequencies (by varying from once up to 4 times every hour). Also in this case, no permanent fault is assumed to have occurred in memory (i.e. $\lambda_e = 0$).

The results show that the RS(18,16) code can not guarantee a low BER (less than 10^{-12}) during solar flares even by using scrubbing. Therefore in this case the use of the RS(36,32) code is mandatory. Using a RS(36,32) code, the scrubbing frequency required for keeping the BER to a value below 10^{-12}, is less than once per hour.

A further evaluation that has been undertaken, consists of assessing the behavior of the RS code in the presence of permanent faults. Fig. 8 shows the results of RS(36,32) under different permanent failure rates. The scrubbing period is given by 1000 sec and the permanent failure rate is in the range of $\lambda_e \in [1 \cdot 10^{-11}, 1 \cdot 10^{-4}]$. In this case the data is assumed to be permanently stored into memory for the whole mission lifetime. Therefore, the results are for a storage time period of about 24 months as mission time. An evaluation of the BER with a storage time of $T_{st} = 48\ h$ as in previous cases, could be performed with suitable modification to the Markov model and by taking into account the additional transitions for $1/T_{st}$.

The scenarios studied in this section are just few examples of the evaluation that can be performed by exploiting the flexibility of the proposed method. In particular, the dependency of BER on different parameters can be pursued to select the architecture for a specific application.

8. Conclusions

This paper has addresses issues related to the efficient design of memory systems for application in critical environments such as avionics. The use of Reed-Solomon codes to obtain high reliability and data integrity has been described in detail; design features and architectural considerations for high performance have also been evaluated. The use of RS codes to detect and locate permanent faults in the memory elements has been analyzed. Different strategies for the detection of permanent faults have been proposed, and a discussion of features has been extensively reported. This analysis focuses on hardware complexity, off-line operation and availability of information for the behavior of memory chips under the occurrence of permanent faults.

An estimate of hardware complexity has been provided for a functional test based detection method; the probability to correctly handle an erasure at the end of

the mission and an architectural solution to reduce the impact of a functional test for off-line time have been proposed.

The analysis of the erasure detection methods (which are related to high level design) has been complemented by an analysis of the various operational parameters and on their impact on the BER of the system. The considered operational parameters are dependent on the physical environment, architectural features (such as scrubbing period or dataword and codeword lengths), as well as high level design choices (such as BER). A method for evaluating the BER has been proposed; this method is based on Markov modeling and allows the evaluation of data integrity with respect to a chosen RS(n,k) code and both permanent and transient faults. In particular, the proposed model takes into account the degradation of data integrity caused by the occurrence of permanent faults while it allows to consider BER as a function of time. Finally, it has been shown that the proposed method permits an extensive flexibility for evaluating different techniques to improve the BER (such as scrubbing and spare memory modules).

References

1. Altera Reed-Solomon compiler User Guide 3.3.3.
2. G. Berger, G. Ryckewaert, R. Harboe-Sorensen, and L. Adams, "Cyclone—A Multipurpose Heavy Ion, Proton and Neutron SEE Test Site," RADECS Radiation and its effects on Components and Systems, 1997.
3. R.E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Company, 1983.
4. G.C. Cardarilli, A. Leandri, P. Marinucci, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Design of a Fault Tolerant Solid State Mass Memory," *IEEE Transactions on Reliability*, vol. 52, no. 4, pp. 476–491, 2003.
5. A. Dinh and D. Teng, "Design of a High-Speed [255, 239] RS Decoder Using 0.18 m CMOS," in *Canadian Conference on Electrical and Computer Engineering*, 2004.
6. J. Fox, W.E. Abare, and A. Ross, "Suitability of COTS IBM 64 Mb DRAM in Space," *Fourth European Conference on Radiation and Its Effects on Components and Systems, RADECS 97*, 1997, pp. 240–244.
7. W. Gao and S. Simmons, "A Study on the VLSI Implementation of ECC for Embedded DRAM," *IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 1, pp. 203–206, 2003.
8. A.H. Johnston, "Radiation Effects in Advanced Microelectronics Technologies," *IEEE Transactions on Nuclear Science*, vol. 45, no. 3, 1339–1354, 1998.
9. Y. Katayama, Y. Negishi, and S. Morioka, "Efficient Error Correction Code Configurations for Quasi-Nonvolatile Data Retention by DRAMs," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 201–209, Oct. 2000.
10. Y. Katayama, E.J. Stuckey, S. Morioka, and Z. Wu, "Fault-Tolerant Refresh Power Reduction of DRAMs for Quasi-Nonvolatile Data Retention," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT '99*, pp. 311–318, Nov. 1999.
11. S. Kwon and H. ShinDept, "An Area-Efficient VLSI Architecture of a Reed-Solomon Decoder/Encoder for Digital VCRs," *IEEE Transactions on Consumer Electronics*, vol. 43, pp. 1019–1027, 1997.
12. P.K. Lala, *Fault Tolerant and Fault Testable Hardware Design*, Prentice-Hall, 1985.
13. H. Lee, "High-Speed VLSI Solomon Decoder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 2, 2003.
14. J. Liu and R.Z. Makki, "SRAM Test Using On-Chip Dynamic Power Supply Current Sensor, Memory Technology," *Design and Testing Proceedings. International Workshop on*, 1998, pp. 57–63.
15. MIL-HDBK 217.
16. C. Paar and M. Rosner, "Comparison of Arithmetic Architectures for Reed-Solomon Decoders in Reconfigurable Hardware," *Symposium on Field-Programmable Custom Computing Machines*, 1997, p. 219–225.
17. H. Pilo, D. Anand, J. Barth, S. Burns, P. Corson, J. Covino, and S. Lamphier, "A 5.6-ns Random Cycle 144-Mb DRAM with 1.4 Gb/s/pin and DDR3-SRAM Interface," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, 2003.
18. B. Polianskikh and Z. Zilic, "Design and Implementation of Error Detection and Correction Circuitry for Multilevel Memory Protection," *IEEE International Symposium on Multiple-Valued Logic, ISMVL 2002*, May 2002, pp. 89–95.
19. A.M. Saleh, J.J. Serrano, and J.H. Patel, "Reliability of Scrubbing Recovery-Techniques for Memory Systems," *IEEE Transactions on Reliability*, vol. 39, pp. 114–122, 1990.
20. L. Schiano, M. Ottavi, and F. Lombardi, "Markov Models of Fault-Tolerant Memory Systems Under SEU," *IEEE International Workshop on Memory Technology, Design and Testing*, 2004.
21. J.M. Soden, "IDDQ Testing for Submicron CMOS IC Technology Qualification, IDDQ Testing, Digest of Papers," *IEEE International Workshop on*, pp. 52–56, 1997.
22. R. Velazco, Ph. Cheynet, A. Bofill, and R. Ecoffet, "THESIC: A Testbed Suitable for the Qualification of Integrated Circuits Devoted to Operate in Harsh Environment," *IEEE European Test Workshop (ETW.98)*, pp. 89–90, 1998.
23. W. Wilhelm, "A New Scalable VLSI Architecture for Reed-Solomon Decoders," *IEEE Journal of Solid-State Circuits*, vol. 34, 1999, 388–396.
24. Xilinx Inc., Content-Addressable Memory V3.0 Data Sheet, Product Specification.
25. Xilinx Inc., Virtex-II Field-Programmable Gate Arrays Data Sheet, Advance Product Specification.
26. Xilinx Logicore Reed-Solomon Decoder v5.1.
27. G.C. Yang, "Reliability of Semiconductor RAMs with Soft-Error Scrubbing Techniques," *Computers and Digital Techniques IEE Proceedings*, vol. 142, no. 5, pp. 337–344, 1995.
28. J.F. Ziegler, "Terrestrial Cosmic Ray Intensities," *IBM J. Res. Develop.*, vol. 42, no. 1, pp. 117–139, 1998.
29. J.F. Ziegler and M.E. Nelson et al., "Cosmic Ray Soft Error rates of 16-Mb DRAM Memory Chips," *IEEE J. Solid-State Circuits*, vol. 33, no. 2, 1998.

Gian Carlo Cardarilli received the Laurea (summa cum laude) in 1981 from the University of Rome La Sapienza. He works for the University of Rome Tor Vergata since 1984. At present he is full professor of Digital Electronics and Electronics for Communication Systems at the University of Rome Tor Vergata. During the years 1992–1994 he worked for the University of L Aquila. During the years 1987/1988 he worked for the Circuits and Systems team at EPFL of Lausanne (Switzerland). Professor Cardarilli interests is in the area of VLSI architectures for Signal Processing and IC design. In this field he published over 140 papers in international journals and conferences. He also participated to the work group of JESSI-SMI for the support to the medium and small industries. For this structure he consulted different SMIs, designing a number ASICs, in order to introduce the microelectronics technology in the industry's products. He has also regular cooperation with companies like Alenia Aerospazio, Rome, Italy, STM, Agrate Brianza, Italy, Micron, Avezzano, Italy, Ericsson Lab, Rome, Italy and with a lot of SMEs. Scientific interests of Professor Cardarilli concern the design of special architectures for signal processing. In particular, he works in the field of computer arithmetic and its application to the design of fast signal digital processor. He also developed mixed-signal neural network architectures implementing them in silicon technology. Recently, he also proposed different new solutions for the implementation of fault-tolerant architectures.

Fabrizio Lombardi graduated in 1977 from the University of Essex (UK) with a B.Sc. (Hons.) in Electronic Engineering. In 1977 he joined the Microwave Research Unit at University College London, where he received the Master in Microwaves and Modern Optics (1978), the Diploma in Microwave Engineering (1978) and the Ph.D. from the University of London in 1982.

He is currently the holder of the International Test Conference (ITC) Endowed Professorship at Northeastern University, Boston. At the same Institution during the period 1998–2004 he served as Chair of the Department of Electrical and Computer Engineering. Prior to Northeastern University he was a faculty member at Texas Tech University, the University of Colorado-Boulder and Texas A&M University.

Dr. Lombardi has received many professional awards: the Visiting Fellowship at the British Columbia Advanced System Institute, University of Victoria, Canada (1988), twice the Texas Experimental Engineering Station Research Fellowship (1991–1992, 1997–1998) the Halliburton Professorship (1995), the Outstanding Engineering Research award at Northeastern University (2004) and an International Research award from the Ministry of Science and Education of Japan (1993–1999). Dr. Lombardi was the recipient of the 1985/86 Research Initiation award from the IEEE/Engineering Foundation and a Silver Quill award from Motorola-Austin (1996).

Dr. Lombardi was an Associate Editor (1996–2000) of IEEE Transactions on Computers and a Distinguished Visitor of the IEEE-CS (1990–1993 and 2001–2004). Since 2000, he has been the Associate Editor-In-Chief of IEEE Transactions on Computers and an Associate Editor of the IEEE Design and Test Magazine. Since 2004 he serves as the Chair of the Committee on "Nanotechnology Devices and Systems" of the Test Technology Technical Council of the IEEE.

Dr. Lombardi has been involved in organizing many international symposia, conferences and workshops sponsored by professional organizations as well as guest editor of Special Issues in archival journals and magazines such as the IEEE Transactions on Computers, IEEE Transactions on Instrumentation and Measurement, the IEEE Micro Magazine and the IEEE Design & Test Magazine. He is the Founding General Chair of the IEEE Symposium on Network Computing and Applications.

His research interests are testing and design of digital systems, quantum and nano computing, ATE systems, configurable/network computing, defect tolerance and CAD VLSI. He has extensively published in these areas and edited six books.

Marco Ottavi is currently postdoctoral research associate at the ECE Department of Northeastern University in Boston. He received the Laurea degree in Electronic Engineering from University of Rome "La Sapienza" in 1999 and the Ph.D. in Microelectronics and Telecommunications from University of Rome "Tor Vergata" in 2004. In 2000 he was with ULISSE Consortium, Rome as designer of digital systems for space applications. In 2003 he was visiting research assistant at ECE Department of Northeastern University. His research interests include yield and reliability modeling, fault-tolerant architectures, on-line testing and design of nano scale circuits and systems.

Salvatore Pontarelli is currently postdoctoral research associate at the University of Rome, Tor Vergata. He received the Laurea degree in Electronic Engineering from the University of Bologna in 1999 and the Ph.D. in Microelectronics and Telecommunications Engineering from the University of Rome Tor Vergata in 2003. His research mainly focuses on fault tolerance, on-line testing and reconfigurable digital architectures.

Adelio Salsano was born in Rome on December 26, 1941 and is currently full professor of Microelectronics at the University of Rome, Tor Vergata where he teaches the courses of Microelectronics and Electronic Programmable Systems. His present research work focuses on the techniques for the design of VLSI circuits, considering both the CAD problems and the architectures for ASIC design. In particular, of relevant interest are the research activities on fault tolerant/fail safe systems for critical environments as space, automotive etc.; on low power systems considering the circuit and architectural points of view; and on fuzzy and neural systems for pattern recognition. An international patent and more than 90 papers on international journals or presented in international meetings are the results of his research activity. At present he is the President of a national consortium named U.L.I.S.S.E., between ten universities, three polytechnics and several of the biggest national industries, as STMicroelectronics, ESAOTE, FINMECCANICA. He is responsible for contracts with the ASI, Italian Space Agency, for the evaluation and use in space environment of COTS circuits and for the definition of new suitable architectures for space applications. Professor Salsano is also involved in professional activities in the field of information technology and is also consultant of many public authorities for specific problems. In particular, he is consultant of the Departments of the Research and of the Industry, of IMI and of other authorities for the evaluation of industrial public and private research projects. Professor Salsano was a member of the consulting Committee for Engineering Sciences of the CNR (National Research Council) from 1981 to 1994 and participated in the design of public research programs in the fields of Telematics, Telemedicine, Office Automation, Telecommunication and, recently, Microelectronics and Bioelectronics.