ROM BASED QUADRUPLE ERROR CORRECTION FOR 4-BIT ADJACENT ERRORS

[1]Dokku Kedarnath Yadav , [2]Dr E JOHN ALEX
[1,2,]Dept Of ECE CMR Institute of Technology (Hyderabad),Telangana , INDIA

Abstract*:*

A standard system-level approach to harden memory against multiple bit upsets (MBUs) is the use of advanced error correction codes (ECCs) correction technologies. Consequently the design of ECCs with enhanced error correction and poor reliability has been an significant problem particularly for adjacent ECCs. Emerging MBU mitigation codes rely mainly on error correction of up to 3-bit bursts.When the device grows and the cell frequency size decreases, the amount of impacted bits quickly expands to more than 3 bit. Consequently, the previous approaches in harsh environments are not adequate to meet the reliability requirements of the applications. A technique is provided in this paperA technique to extend the 3-bit burst error correction (BEC) codes with quadruple adjacent error correction (QAEC). Initially, the principles of the interface are established, bur now a search algorithm is built to find the codes which match these principles. The 3-bit matrices obtained for BEC H come with QAEC. They will not require extensive check bits for consistency relative to a 3-bit BEC register. The performance of the 3-bit BEC is also considerably increased when adding the latest algorithm to previous 3-bit BEC files.

Significant words:

*B*urst error-correction codes (BECCs), multiple bit upset (MBU), memory, quadruple adjacent error correction (QAEC).

I. INTRODUCTORY

Error correction codes are commonly used to protect memories through so-called soft errors, that alter the memory cell 's functionality without damaging the circuit. As technology advances, memory modules are bigger, needing very powerful error correction codes. It has been proposed recently that more complex codes should be used for the above reason. Such codes may contain additional errors usually involving complicated codecs.

The usage with one-step majority logic decodable codes for memory applications had been introduced to avoid a significant decoding complexity.

Yet another phase majority logic decoding with quite simple circuitry can be done serially but it requires long decoding cycles. It will boost the time taken to reach a script. Using OS-MLD that you can only decode a few forms of codes. Several of these contain codes DSLDPC, codes EG-LDPC, and codes OLS.The usage of OLS codes has created revived curiosity in interconnections, images, and caching.Its feature enables it easy to easily adapt the error correction function to the error rate or operating mode. OLS codes normally require additional redundancy pieces to rectify the same amount of errors as other protocols.

Its modularity and the simple and quick delay implementation of the decoding overcome this downside (as OLS codes are OS-In other MLD deployments). A big concern is that the related encoder and decoder circuits (ECCs) will also suffer from errors.

If a error hits the encoder, an erroneous word may be added to the log. A decoder error can result in a right word being interpreted as incorrect or the other way around, a wrong word being perceived as the perfect description.

A technique to speed up sequential execution of encoding of DS-LDPC codes via majority logic has recently been proposed.

The idea behind the technique is to evaluate if the word being decoded includes defects, utilizing the first iterations of decoding majority reasoning. When no errors occur, the encoding can be stopped until the remaining iterations have been done,That dramatically reduces decoding time.

Even with basic equipment, If a error hits the encoder, an erroneous word may be added to the log. A decoder error can result in a right word being interpreted as incorrect or the other way around, a wrong word being perceived as the perfect description.

II. Existing system

When technology grows, memory cards become wider and more efficient error correction codes become required. Recently it was proposed that more advanced codes would be used to this purpose. These codes can fix additional errors which would normally need complex decoders. Use of such one-step majority logic decodable codes was originally implemented for memory applications to avoid a strong decoding burden. And further work on this subject was dealt with in. One-step majority logic decoding with very easy circuitry can be applied in series but it requires long decoding periods. It will increase the access time in a memory and is a significant parameter for applications.

Only a few kinds of codes can be decoded using the encoding of a single phase majority logic. Certain codes contained in Euclidean Low Density Parity Check (EG-LDPC) and Low Density Parity Check (DS-LDPC) codes that set contradictions, are among them.

This approach was adopted by majority logic to speed up the decoding of variations by defining the search codes for low density parity. It is helpful since with simple hardware most logic decoding can be achieved serially, but decoding takes substantial time. It reduces the time needed to reach computers using Ram.

The process decides whether a term has errors during the first iterations of majority logic decoding, but if there are no errors, the decoding ends without the rest of the iterations. Since several of the words in a brain are error-free, the overall encoding time is considerably reduced. In this summary, we are discussing the application of a similar methodology to the Euclidean Low Density Parity Check (EG-LDPC) codes type, which are decodable by majority logic. The results obtained suggest that the protocol for the EG-LDPC codes is also correct.

Statistics are given to correctly measure the possibility of finding errors on various code sizes and error amounts.

A technique has recently been proposed to speed up sequential encoding of DS-LDPC codes via majority logic. The principle behind the method is to determine how the term being decoded involves errors, utilising the first iterations of consensus logic to decipher. If no errors occur, it is possible to stop decoding instead without completing the remaining iterations which significantly reduces the decoding time.

Largest logic decoding (when serially implemented) requires N iterations for a record with block length N, such that the decoding period decreases as the file size decreases. Under the new method, only the first three versions are used to identify mistakes, resulting in a significant timely change.

It has been shown to recognise these error combinations of up to 5 errors in the first three iterations for DS-LDPC codes. Errors of more than 5 bits were often detected with an extremely high chance of one. The probability of undetected errors was also noted as the code block length increased.

A few failures of one billion errors (or at times none) are not observed.

For some programmes, this may be appropriate. The downside is that very little additional circuitry is required, as the decoding circuit is used to detect errors already. For example , the increased area required to affect small word sizes was shown

You will sequentially extend the scheme in Fig 3.1 that corresponds to the one-step Example LDPC MLD decoder with N=15. Then you load in the registers a storage table. The control equations are computed and the last bit reversed if most have one value.

Each element is therefore moved cyclically.

This series reflects a single iteration: after N iterations, the bits remain in the same position. Any bit can only be changed once during the whole process. The decoding circuit is as simple as shown, but decoding is long because Nis is large.

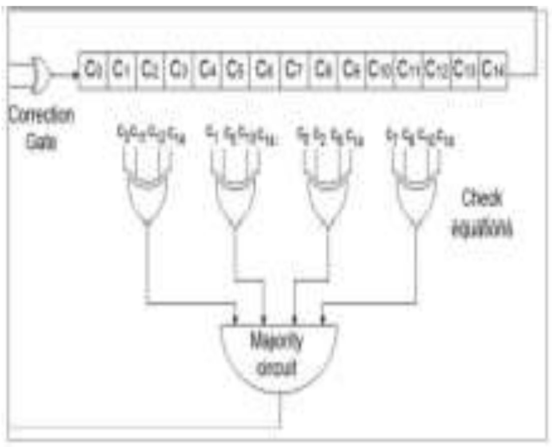


Fig:1 Serial one-step majority logic decoder for the EG-LPDC code.

The control equations contain the following characteristics 1. -- calculation requires the element, the value of which is stored in the last register (C14).

2. Up to one of the monitoring steps is carried out in the remaining registers.

If errors in the first few iterations of the MLD can be observed, no errors in those iterations are found next time. Decoding can be stopped without the current versions being lost. In the initial run, errors will be noticed when at least one of the check measurements reaches an unexpected error number of bits. In the second case, if the bits are rotated cyclically by one place in the first iteration, errors will impact some steps such that any undetected errors are noticed. Basically, as iterations continue, all measurable errors are observed.

It was shown that in the first three iterations of MLD, various errors are present in DSLDPC codes. In the case of two failures, a modelling analysis and a potential proof were considered to be the following conclusion. The proposed technique was then applied and synthesised in Verilog, demonstrating that the overhead for large block-sized codes is small. That is because the existing decoding circuit was reused to allow the error detection for certain lots. "A memory word that is covered for DS-LDPC codes and inspired by up to five bitflips can be found in three decoding cycles only."

III. PROPOSED ARCHITECTURE IMPLEMENTATION

The OLS codes focus on Latin squares interpretation. In both rows and columns with digit permutations, the Latin square of size m is the tuple of 1, ..., m − 1. If every ordered pair of elements exists only once when superimposed, two latino squares are orthogonal.

Such codes have k = m2 and 2tm check bits, of which t is the number of errors which the programme can overcome. For t=2 and 4 m test bits, a double error correction code.

One benefit of OLS codes is that they are flexible in the design, as stated in the introduction.

This means that 2 m test bits are clearly applied to the code to produce a code that can fix t + 1 errors.It can be helpful when designing proactive error correcting systems. The modular property also requires a defined word size to be selected for the error correction power. Using OS-MLD, OLS codes can be decoded because each data bit is involved in exactly 2 t test bits and at most one of those search bits is involved in each other part. If the amount of error bits is t or less it allows for a fast patch. The sections of the 2 t check are reputed and require a plurality ballot. When the interest of one is obtained the bit is in error and has to be changed.

Therefore, the part is wrong. In the worst case, the residual t −1 errors would affect t −1 check bits, assuming that the amount of errors is t or less.

$$H=\begin{bmatrix}
1&0&0&0&0&0&0&1&0&0&1&0&1&0&1&1&1&0&0&0&1&0&0\\
0&1&0&0&0&0&0&0&1&0&1&1&0&1&1&0&0&1&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&1&1&0&1&0&0&1\\
0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&1&0&1&1&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&1&1&1&1&0&0&0&0&1&1&0&0\\
0&0&0&0&0&1&0&0&1&0&0&0&1&1&0&0&0&1&0&1&0&0&1\\
0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&1&0&1&1&1&0&1&1
\end{bmatrix}$$

Fig:2 Parity check matrix for OLS code with *k* = 16 and *t* = 1.

(1) While the remainder of t + 1 still triggers erroneous bit correction. In any case, the decoding starts by recalculating the check bits for parity and

Test the parity testing bits held. For the OLS symbols, the parity test matrix H is built from the OLS.

The matrix of k= 16 and 8 control bits that can fix individual errors is shown in the figure. 1. In this matrix, the modular design of OLS codes forms part of the H matrix and can correct more mistakes. For example, eight more rows of the H matrix are applied to achieve a code that can fix two errors. The matrix H for an SEC OLS code is generated as follows for an arbitrary value of k= m2:

$$H = \begin{bmatrix} M_1 & \\ & I_{2m} \\ M_2 & \end{bmatrix}$$

Where I2 m is 2 m and M1 matrix, M2 is a m= m2 matrix. Every segment of the M1 matrix has m.

The positions

$$r-1)=m+1, (r-1)=m+2, \ldots (r-1)=m+m-1,$$
$$(r-1)=m+m$$

for the

r th lines. The M2 matrix is constructed as follows:

M2=[I m I m.] I m]. (2)

The matrices M1 and M2 can be clearly seen in Figure for m=4. 1. The matrix encoding G is just the vector H which eliminates the test bits.

$$G = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}.$$

In brief, the encoder takes k = m2 data bits (di) and uses a matrix G constructed from Latin squares to calculate 2tm parity test bits (ci) with the following properties.

1) The 2 t parity checks are used specifically for each data bit.

2) At most, one of the parity tests involves a pair of bits of data (both bits).

Those properties are included in the next segment to illustrate the suggested technique.

IV. Suggested Methodology for Parallel Error Detection

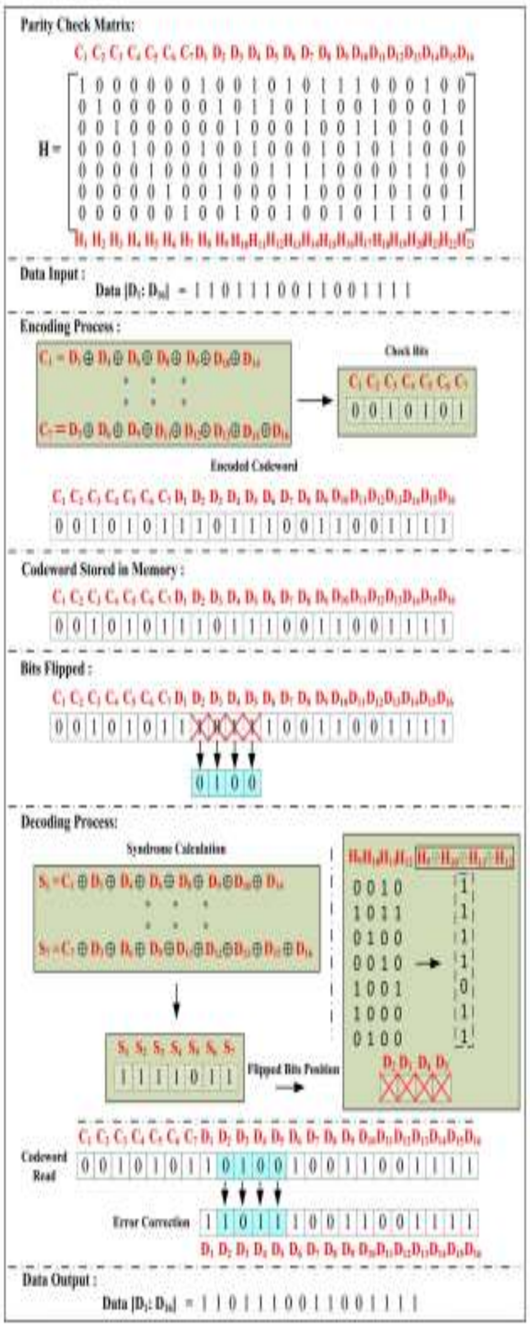
The check bits are determined by the following data bits based on the parity control matrix description. The current binary codeword, checkbits and app bits are contained in the memory. The material of damaged memory cells are reverted after the ions enter the memory and result in MBUs. In D2, D3, D4 and D5, four adjacent bits are flipped to extend the clarifiers of QAEC codes. The syndrome is measured using stored check bits and data bits as well as the parity search matrix form in the decodification process.

The corresponding link between the syndrome and the column XOR element referred to in Part II

The brain retention stage fault is effectively reversed for the flipped bits twisted. It is the entire way to encrypt and decode modern QAEC codes.

An extension of the QAEC 3-bit BEC codes is given. The codes were out of line for the current 3 bit BEC codes. The coding was out of line. Advanced weight control and algorithms for the screening of target matrices are proposed to speed up.

The suggested system could eventually be extended to a 3-bit ECC explosion method which could address a specific 4-bit exploding error pattern instead of an identical 4-bit error pattern. This could be interesting for programs without an associated 4-bit sequence of destructive errors. The overhead time and space of the previous 3-bit BEC code is low.



EXTENSION OF PROPOSED SYSTEM:

If S0i is a single bit error syndrome S1i is a double neighboring bit error syndrome, S2i is a three-bit adjacent bit error, S3i is a 3-bit closely overlapping bit error syndrome and S4i is a quadruple bit loss syndrome adjacent to it. S4i is the syndrome of double bit error. The state parameters of S0i, S1i, S2i, S3i and S4i are linear combinations of H matrix columns according to the rules.

$S0i = hi$

$S1i = hi \oplus hi-1$

$S2i = hi \oplus hi-1 \oplus hi-2$ (12) $S3i = hi \oplus hi-2$

$S4i = hi \oplus hi-1 \oplus hi-2 \oplus hi-3$

As described above, in 4bit error cases we will measure 20 syndromes for each data source. We can use these syndromes any time, that's why we're going to store these syndromes in ROM.

Step1: received error vector

Step2: calculate syndrome with the help of H matrix

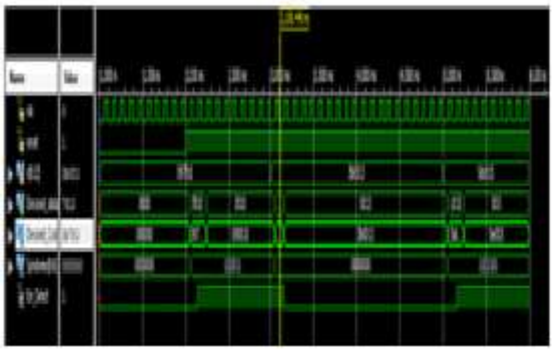
Step3:if syndrome is zero then there is no error present ,if syndrome is non zero then go to memory .check syndromes (as all syndromes are unique)we will come to know where is the error mean at which position error has occurred.

Step4: flip those bits error the received vector.

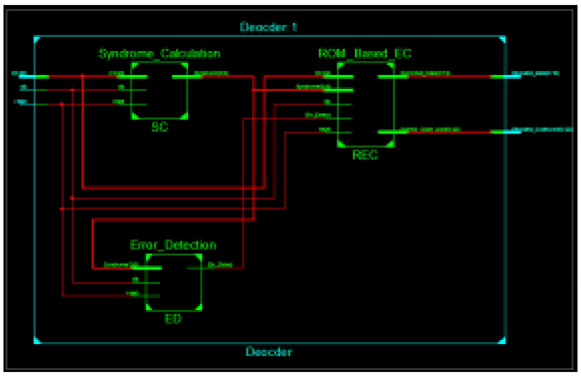
VI. SIMULATION RESULTS

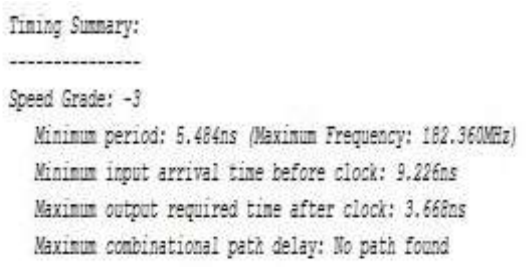
PROPOSED EXTENSION SYSTEM RESULTS:

simulation result:



RTL SCHEMATIC:

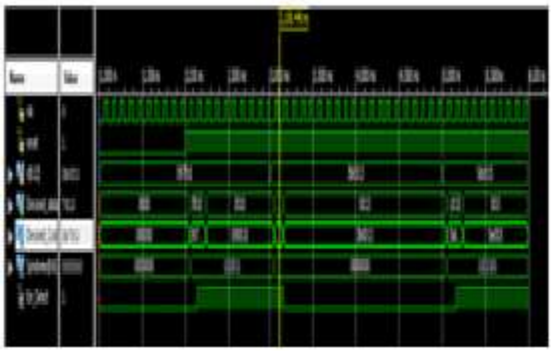




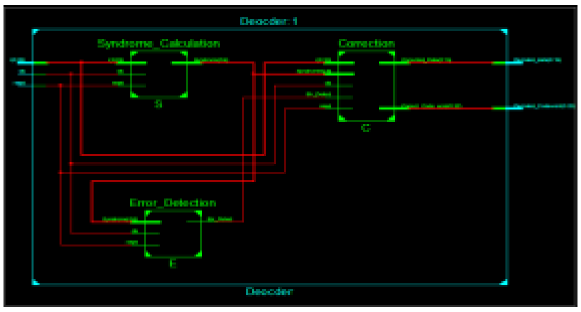
```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 5.484ns (Maximum Frequency: 182.360MHz)
   Minimum input arrival time before clock: 9.226ns
   Maximum output required time after clock: 3.668ns
   Maximum combinational path delay: No path found
```

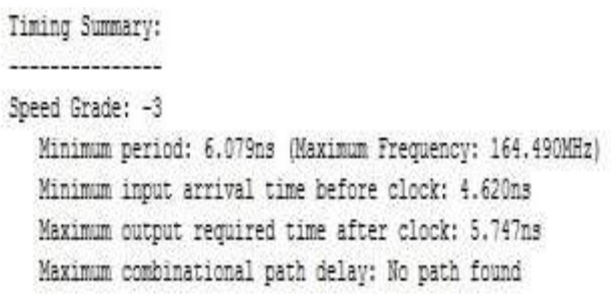
Proposed system:

Simulation result:



RTL SCHEMATIC:





```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 6.079ns (Maximum Frequency: 164.490MHz)
   Minimum input arrival time before clock: 4.620ns
   Maximum output required time after clock: 5.747ns
   Maximum combinational path delay: No path found
```

COMPARISION TABLE:

S.No	QAEC	delay
1	Extension	5.484ns
2	Proposed	6.079ns

VII. CONCLUSION

The CED platform was proposed for OLS coders and functional syndrome encoders. With the property of OLS codes, the proposed technology identifies all errors that include a single circuit node and establishes a readily implementable parity prediction scheme.

The methodology was examined with different word size and showed that the overhead is small in broad terms. For examples, the large word sizes are used in caches that recently proposed OLS codes This is necessary

The suggested error detection scheme involved a significant delay; however, the effect on the access time could be that. In the case of the encoder, parallel testing of the majority vote and correction of an error was done before the results is reported and the decoder was reported. In addition, the new scheme demanded a lot greater overhead, as the majority of ECCs had no OLS code capital. It limited the use of the CED system for OLS codes.

The use of low overhead error detecting technique for encoders is another justification that can be used to recommend OLS codes in high-speed memorization. The use of ols in high-speed processing and retrieval can be a further motivation.

REFERENCES

[1] R. C. Baumann, “Radiationinduced soft errors in advanced semiconductor technologies,” *IEEE Trans. Device Mater. Reliab.*, vol. 5, no. 3, pp. 301–316, Sep. 2005.

[2] M. A. Bajura, Y. Boulghassoul, R. Naseer, S. DasGupta, A. F.Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill, and J. N. Damoulakis, “Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs,” *IEEE Trans. Nucl. Sci.*, vol. 54, no. 4, pp. 935–945, Aug. 2007.

[3] R. Naseer and J. Draper, “DEC ECC design to improve memory reliability in sub-100 nm technologies,” *Proc. IEEE ICECS*, pp. 586–589, 2008.

[4] S. Ghosh and P. D. Lincoln, “Dynamic low-density parity check codes for fault tolerant nano-scale memory,” presented at the Foundations Nanosci. (FNANO), Snowbird, Utah, 2007.

[5] S. Ghosh and P. D. Lincoln, “Lowdensity parity check codes for error correction in nanoscale memory,” SRI Computer Science Lab., Menlo Park, CA, Tech. Rep. CSL-0703, 2007.

[6] H. Naeimi and A. DeHon, “Fault secure encoder and decoder for memory applications,” in *Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Syst.*, 2007, pp. 409–417.

[7] B. Vasic and S. K. Chilappagari, “An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on lowdensity parity-check codes,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.

[8] H. Naeimi and A. DeHon, “Fault secure encoder and decoder for nanomemory applications,”*IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 4, pp. 473–486, Apr. 2009.

[9] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.

[10] S. Liu, P. Reviriego, and J. Maestro, “Efficient majority logic fault detection with difference-set codes for memory applications,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 148–156, Jan. 2012.

[11] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, “Codes on finite geometries,” *IEEE Trans. Inf. Theory*, vol. 51, no. 2, pp. 572–596, Feb. 2005.

[12] R. D. Schrimpf and D. M. Fleetwood, Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices. Singapore: World Scientific, 2004.

[13] S. K. Vishvakarma, B. S. Reniwal, V. Sharma, C. B. Khuswah, and D. Dwivedi, “Nanoscale memory design for efficient computation: Trends, challenges and opportunity,” in Proc. IEEE Int. Syst. Nanoelectron. Inf. Syst., Dec. 2015, pp. 29–34.

[14] S.-Y. Wu, C. Y. Lin, S. H. Yang, J. J. Liaw, and J. Y. Cheng, “Advancing foundry technology with scaling and innovations,” in Proc. Int. Symp. VLSI-TSA, Apr. 2014, pp. 1–3.

[15] ITRS. International Technology Roadmap for Semiconductors (ITRS) Report, Semiconductor Industry Association. Accessed: 2009. [Online]. Available: http://www.itrs.net

[16] Y. Bentoutou, “A real time EDAC system for applications onboard earth observation small satellites,” IEEE Trans. Aerosp. Electron. Syst., vol. 48, no. 1, pp. 648–657, Jan. 2012.

[17] E. Ibe, H. Taniguchi, Y. Yahagi, K.-I. Shimbo, and T. Toba, “Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule,” IEEE Trans. Electron Devices, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[18] L. Artola, M. Gaillardin, G. Hubert, M. Raine, and P. Paillet, “Modeling single event transients in advanced devices and ICs,” IEEE Trans. Nucl. Sci., vol. 62, no. 4, pp. 1528–1539, Aug. 2015.

[19] M. Murat, A. Akkerman, and J. Barak, “Electron and ion tracks in silicon: Spatial and temporal evolution,” IEEE Trans. Nucl. Sci., vol. 55, no. 6, pp. 3046–3054, Dec. 2008