The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories

Dmitri Strukov
Stony Brook University
Stony Brook, NY 11794-2350 USA
Email: dstrukov@ic.sunysb.edu

***Abstract*— We have investigated the area and latency tradeoffs with respect to error correcting capability of fast bit-parallel binary BCH ECC decoders. In particular, we show that for a primitive BCH code of length n over GF(2^m) with a Hamming distance of at least $2t+1$ the area and latency scale approximately as nm^2t and mt, respectively. The results presented in this paper might be very useful, e.g., for assessing the performance overheads due to the ECC decoders in the future random access nanoelectronic memories, which are expected to have a significantly larger number of defects as compared to that of today's CMOS memories.**

I. Introduction

The recent spectacular progress in molecular electronics (for reviews see, e.g., Ref. [1], and, in particular, demonstration of molecular-scale nanodevices with characteristics plausible for memory applications [2]–[4] give hope for practical introduction of this technology in the near future. In nanoelectronic memories, e.g., CMOL memories [5]–[7], bits can be implemented with nanodevices which are sandwiched in between two layers of nanowire crossbar, while the other circuitry (e.g., drivers, sense amplifiers, decoders, and error-correcting circuitry) is implemented in CMOS subsystem. The first experimental steps toward the implementation of such hybrid memories have already been made; see, for example, Refs. [8], [9].

Since the most plausible way to integrate nanodevices into the large scale circuit is through some kind of self-assembly process, the number of defects (or more specifically the failures to self assembly of nanodevices on the crosspoints of the nanowire crossbar) will probably never be as small as that of the CMOS memories, and the defect rates of up to few percent are very likely. (For example, 15% of the bits in the experiment in Ref. [8] were defective.) Therefore, conventional techniques like matrix reconfiguration which are used in the CMOS memories to cope with static defects will not be sufficient and should be synergistically used with error-correcting codes (ECC) [6], [7], [10], [11].

The ECCs are being used extensively in virtually all commercial memory and storage devices to tolerate both transient faults and static errors. Typically, light ECCs (e.g., Hamming codes) are used for fast memories, like SRAMs and DRAMs, and more powerful ECCs (e.g., Reed Solomon codes) are used in flash memories and magnetic/optical storages devices. Since the latter have intrinsically slow access time much of the research was done for optimizing the throughput of the ECC decoders rather than their latency [12], [13]. On the contrary, the access time of hybrid nanoelectronic memories can be very small, i.e. in the nanosecond range [7], the fact that makes them suitable for random-access operations. Hence, it is advantageous to use both high-speed and powerful ECCs which can cope with high defect rates presented by this new technology.

Low-latency decoding can be achieved by parallelizing the decoding algorithm. One obvious limitation for parallelization is the increase of the ECC decoder area. For example, standard array decoding [14], i.e. keeping the memory with leader cosets, would be the fastest technique for decoding linear block codes, however the area of such decoder grows exponentially with the number of errors the code can fix. Therefore, to avoid unpractical solutions, we are interested both in latency and area dependence on error-correcting capability of the code. Similar tradeoffs analysis was carried out for various high throughput ECC decoders [15], [16].

A fast bit-parallel finite field arithmetic (finite field arithmetic is required in many decoding algorithms) was already heavily studied in the context of cryptography [17], [18]. In this paper we use some of these results to study the area and latency tradeoffs of the fast decoders for binary BCH codes.[1] (The binary codes are better suited for our defect model with errors uniformly distributed throughout the memory matrix. Such model is adequate for molecular-scale memories and was widely used in other research papers [11], [19], [20].)

II. Model

A. General Structure

For simplicity, let us consider primitive BCH codes, with the code length on the finite field GF(2^m) is $n = 2^m - 1$. A plausible implementation of the fast decoding for binary BCH codes requires three major steps [13], [14]:

- Step 1: syndrome evaluation;

[1]Though BCH codes are perhaps not the absolutely best option for nanoelectronic memories, they are one of the most efficient among short (less than 1024 bits long) codes. Currently, we are investigating a more natural codes, e.g., Euclidean geometry and LDPC ones [13], [14], and also codes which take advantage of the asymmetry of the error model.

- Step 2: finding the error-location polynomial using the Berlekamp-Massey iterative algorithm; and
- Step 3: finding error-location numbers with subsequent error-correction.

Let us estimate the area and delay contributions of the circuits of each step.

B. Step 1: Syndrome Calculation

In a completely bit-parallel implementation each bit of a syndrome $\boldsymbol{S}$ can be obtained by a separate XOR tree with inputs taken from the received code vector [13]:

$$\boldsymbol{S} = (S_1, S_2, ..., S_{2t}) = \boldsymbol{r}\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{r} \begin{pmatrix} 1 & 1 & . & 1 \\ (\alpha) & (\alpha^2) & . & (\alpha^{2t}) \\ . & . & . & . \\ (\alpha)^{n-1} & (\alpha^2)^{n-1} & . & (\alpha^{2t})^{n-1} \end{pmatrix}. \quad (1)$$

Here $\boldsymbol{r}$ is a received code word of length n, $\mathbf{H}$ is a parity matrix, and α is a primitive element in GF(2^m), i.e. each column in the matrix in the Eq. 1 can be rewritten as an m binary columns. On the average, binary columns of $\boldsymbol{H}$ are half filled with ones while in the worst case there is a column with almost all ones. Hence the syndrome calculation circuit will be comprised of total $2tm$ XOR trees with the average depth $\log_2(n/2)$ and the worst depth of $\log_2 n$ (Fig. 1).

$$\tau_1 = \tau_{\text{add}}(n) = \lceil \log_2(n) \rceil \times \tau_{\text{XOR}}, \quad (2)$$

$$A_1 = 2tm \times A_{\text{add}}(n/2) = tnm \times A_{\text{XOR}}. \quad (3)$$

Here by $\tau_{\text{add}}(n)$ and $A_{\text{add}}(n)$ we denote, correspondingly, the delay and area of 1-bit parallel addition in finite field (XOR tree) of n elements. Similarly, τ_{XOR} and A_{XOR} (and also τ_{OR} and A_{OR} which are used later in text) denote the delay and area of Boolean logic 2-input XOR (OR) gate, respectively.

C. Step 2: Finding Error-Location Polynomial with Berlekamp-Massey Algorithm

Since the algorithm is based on a recursive procedure it cannot be parallelized completely. However, it is possible to speed up the computation of each iteration. Two most time-consuming operations in this step are calculating discrepancy d (element of the field GF(2^m)) and adjusting error location polynomial $\sigma(X)$ if necessary [13], e.g., for the $\mu + 1$ step ($0 \le \mu \le t-1$):

$$\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_\mu d_\rho^{-1} X^{2(\mu-\rho)} \sigma^{(\rho)}(X), \quad (4)$$

$$d_{\mu+1} = S_{2\mu+3} + \sigma_1^{(\mu+1)} S_{2\mu+2} + \sigma_2^{(\mu+1)} S_{2\mu+1} + ... + \sigma_{l_{\mu+1}}^{(\mu+1)} S_{2\mu+3-l_{\mu+1}}. \quad (5)$$

Here $\rho \le \mu$ and $l \le \mu$ are indices specific to the implementation of the algorithm [13]. For the worst case, one needs to update $\sigma_\mu(X)$ in each iteration μ, while the degree of $\sigma_\mu(X)$ exactly equal to μ. Hence the calculation of $\sigma_\mu(X)$ in early iterations can be done with smaller circuit area and (slightly) faster time. However, it also implies different hardware for each iteration. Instead, we will assume the scheme shown in Fig. 2 where the hardware is shared among the iterations and the latency is the same in all iterations, being dictated by the operations on $\sigma_\mu(X)$ with the maximum degree t.

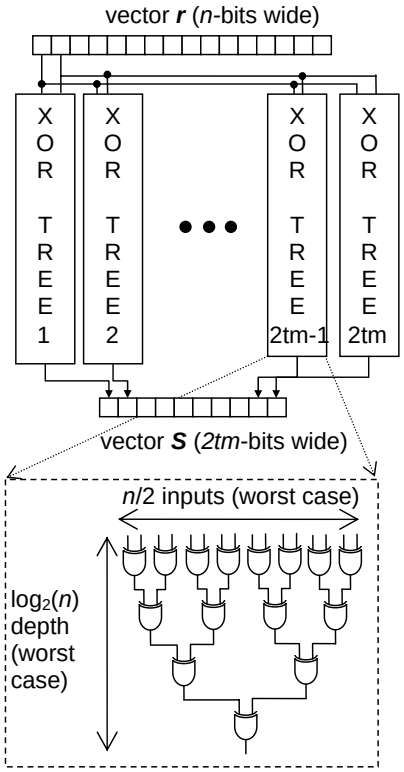

Fig. 1. Syndrome calculation.

Implementation of Eq. 4 (see part 2A on Figure 2) involves one inversion in GF(2^m), $2t$ multiplications in GF(2^m), m additions of t bits and multiplying by the indeterminate $X^{2(\mu-\rho)}$ (i.e. a shift operation). Fast inversion can be done directly as a hard-wired LUT table for small m, so that the complexity of such operation is roughly [17]:

$$\tau_{\text{inv}} = \lceil \log_2(m-1) \rceil \times \tau_{\text{AND}} + (m-1) \times \tau_{\text{OR}}, \quad (6)$$

$$A_{\text{inv}} = \frac{1}{4} m^2 n \times A_{\text{AND}} + \frac{1}{4} mn \times A_{\text{OR}}. \quad (7)$$

For small values of m (less than 10) the Mastrovito multiplier [21] ensures fast efficient implementation comparable with the other approaches [17]. The complexity of such multiplier is

$$\tau_{\text{mult}} = \tau_{\text{AND}} + 2\lceil \log_2 m \rceil \times \tau_{\text{XOR}}, \quad (8)$$

$$A_{\text{mult}} = m^2 \times A_{\text{XOR}} + m^2 \times A_{\text{AND}}. \quad (9)$$

Complexity for the fast shift operation (Fig. 2) is

$$\tau_{\text{shift}} = \tau_{\text{AND}} + \lceil \log_2 m \rceil \times \tau_{\text{OR}}, \quad (10)$$

$$A_{\text{shift}} = mt^2 \times A_{\text{AND}} + mt^2 \times A_{\text{OR}}. \quad (11)$$

Thus, the complexity of the part 2A during one iteration is

$$\tau_{2\text{A}} = \tau_{\text{inv}} + \tau_{\text{mult}} + \tau_{\text{add}} = (1 + \lceil \log_2(m-1) \rceil) \times \tau_{\text{AND}} + (2\lceil \log_2 m \rceil + \lceil \log_2 t \rceil) \times \tau_{\text{XOR}} + (m-1) \times \tau_{\text{OR}}, \quad (12)$$

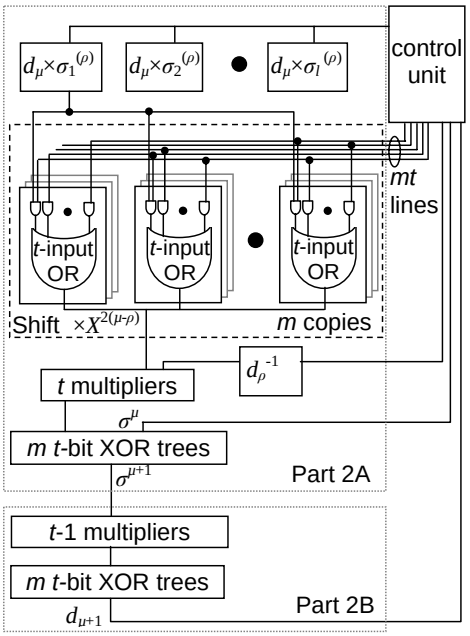

Fig. 2. Finding Error-Location Polynomial with Berlekamp-Massey Algorithm.

$$A_{2A} = A_{inv} + 2t \times A_{mult} + m \times A_{add} = (2mt^2 + 2m^2t + \frac{1}{4}m^2n) \times A_{AND} + (m^2t + mt) \times A_{XOR} + (m^2t + \frac{1}{4}mn) \times A_{OR}. \quad (13)$$

Here it is assumed that inversion is the slower operation than the first multiplication and shift, and hence it is included in the critical path delay calculation. Finally, assuming that part 2B is implemented with $(t-1)$ multiplications and additions, and neglecting other circuitry (e.g., a control unit), the total complexity of the Step 2 is

$$\tau_2 = t(2 + \lceil \log_2(m-1) \rceil) \times \tau_{AND} + t(4\lceil \log_2 m \rceil + 2\lceil \log_2 t \rceil) \times \tau_{XOR} + t(m-1) \times \tau_{OR}, \quad (14)$$

$$A_2 = (2mt^2 + 3m^2t + \frac{1}{4}m^2n - m^2) \times A_{AND} + (3m^2t + 2mt - m^2) \times A_{XOR} + (m^2t + \frac{1}{4}mn) \times A_{OR}. \quad (15)$$

D. Finding Error Location Numbers and Correction

The simple substitution which is performed in parallel for all 2^m elements is the fastest (though rather area-expensive) way to find the error location numbers. The test circuit for checking whether some element from GF(2^m) is a root of error location polynomial $\sigma(X)$ requires a multiplication by a constant (Fig. 3) which has a complexity [17], [21]:

$$\tau_{const} = \lceil \log_2 m \rceil \times \tau_{XOR}, \quad (16)$$

$$A_{const} = (\frac{1}{2}m^2 - m) \times A_{XOR}. \quad (17)$$

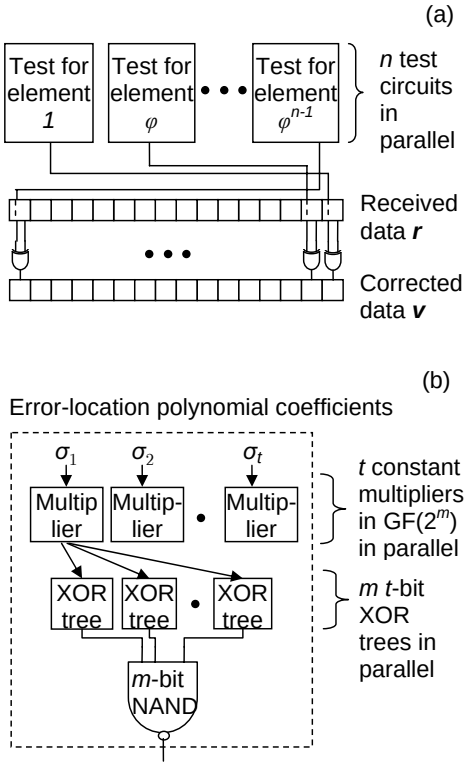

Fig. 3. (a) Finding error location numbers and (b) error correction.

Hence the test circuit can be implemented with

$$\tau_{test} = \tau_{const} + \tau_{add} + \lceil \log_2 m \rceil) \times \tau_{AND} = (\lceil \log_2 m \rceil + \lceil \log_2 t \rceil) \times \tau_{XOR} \lceil \log_2 m \rceil) \times \tau_{AND}, \quad (18)$$

$$A_{test} = t \times A_{const} + m \times A_{add} + m \times A_{AND} = \frac{1}{2}tm^2 \times A_{XOR} + m \times A_{AND}, \quad (19)$$

so that the total complexity of the Step 3 is

$$\tau_3 = \tau_{test} + \tau_{XOR} = \lceil \log_2 m \rceil) \times \tau_{AND} + (1 + \lceil \log_2 m \rceil + \lceil \log_2 t \rceil) \times \tau_{XOR}, \quad (20)$$

$$A_3 = n \times A_{test} + m \times A_{XOR} = (\frac{1}{2}tm^2n + n) \times A_{XOR} + nm \times A_{AND}. \quad (21)$$

III. Results and Discussion

Table I outlines the complexity of key operations used in BCH decoder model, while Table II presents our main results. In particular, the latter shows that for large values of n the area of the decoder scales approximately as nm^2t and mostly dominated by the circuitry in Step 3. On the other hand, most of the delay, which is roughly proportional to mt, comes from Step 2.

As expected the full delay is certainly much better than that of bit-serial BCH decoder [14]. Moreover, Table II shows that it might be possible to reduce the area-delay product without area and delay degradation by balancing performance among all steps. For example, since the delay of the Step 3 is

Operation in GF(2^m)	Area			Critical Path Delay		
	OR	AND	XOR	OR	AND	XOR
Addition (n elements)	0	0	mn	0	0	$\lceil \log_2 n \rceil$
Multiplication	0	m^2	m^2	0	1	$2\lceil \log_2 m \rceil$
Constant Multiplication	0	0	$m^2/2$-m	0	0	$\lceil \log_2 m \rceil$
Inversion	$mn/4$	$m^2n/4$	0	m-1	$\lceil \log_2(m\text{-}1) \rceil$	0
Multiplication by indeterminate of polynomial with degree t	mt^2	mt^2	0	$\lceil \log_2 m \rceil$	1	0

TABLE I
COMPLEXITY OF KEY OPERATIONS IN BCH DECODER

Operation	Area			Critical Path Delay		
	OR	AND	XOR	OR	AND	XOR
Step 1	0	0	tnm	0	0	$\lceil \log_2 n \rceil$
Step 2	$mt^2 + mn/4$	$3m^2t + 2mt^2 + m^2n/4 - m^2$	$3m^2t + 2tm - m^2$	$t(m\text{-}1)$	$2t + t\lceil \log_2(m\text{-}1) \rceil$	$4t\lceil \log_2 m \rceil + 2t\lceil \log_2 t \rceil$
Step 3	0	mn	$m^2tn/2 + n$	0	$\lceil \log_2 m \rceil$	$1+\lceil \log_2 m \rceil + \lceil \log_2 t \rceil$
Total	$mt^2 + mn/4$	$3m^2t + 2mt^2 + m^2n/4 + mn - m^2$	$tnm + 3m^2t + 2tm + m^2tn/2 + n - m^2$	$t(m\text{-}1)$	$2t + t\lceil \log_2(m\text{-}1) \rceil + \lceil \log_2 m \rceil$	$1+\lceil \log_2 n \rceil + (4t+1)\lceil \log_2 m \rceil + (2t+1)\lceil \log_2 t \rceil$

TABLE II
COMPLEXITY OF PROPOSED BCH DECODER

negligible its area might be reduced by implementing slower algorithm.

It is more convenient to express the decoder complexity in technology-independent units such as CMOS half-pitch $F_{\rm CMOS}$ and fanout-of-4 inverter delay $\tau_{\rm FO4}$. To simplify the analysis let us assume the static CMOS gate implementation. Using SCMOS rules [23], the areas of the static minimum-size 2-input OR (6-transistor), 2-input AND (6-transistor) and 2-input XOR (10-transistor) gates are roughly equal to 150, 150 and 250 $(F_{\rm CMOS})^2$, while their delays (assuming that each gate drives only one copy of itself) are about 2/3, 2/3 and 9/5, respectively [24]. Therefore, adding contributions from each step, and droping negligibly small terms, the full area and latency of the BCH decoder can be described by

$$A_{\rm BCH} \approx 125nt(\log_2 n)^2 + 40n(\log_2 n)^2 + 250nt\log_2 n + 190n\log_2 n + 1200t(\log_2 n)^2 + 300t^2\log_2 n. \quad (22)$$

$$\tau_{\rm BCH} \approx 0.7t\log_2 n + 0.7t + 8t\log_2\log_2 n + 3.6t\log_2 t + 2.5\log_2\log_2 n + 1.8\log_2 t + 1.8. \quad (23)$$

Figure 4 shows the dependence of the area and delay versus the code efficiency, i.e. the maximum number of errors t which can be decoded from the minimum distance, for several codes of different block length. We assumed here that each code has mt parity-check bits and the minimum distance of the code is equal to $2t + 1$. (For large values of n and t the number of parity bits can be somewhat smaller [14]; however it does not affect the results on Figure 4.) To appreciate the results for a $F_{\rm CMOS}$ = 22 nm CMOS technology node ($\tau_{\rm FO4}$ = 9 ps) Figure 4 also shows the delay of 1 ns, and also the areas of 0.1 cm^2 and 0.1 mm^2, which may be a reasonable chip's real estate for the stand alone and embedded memories.

It is worth mentioning that the results in Eqs. 23, 22, and also in Figure 4, are rather crude because:

(i) We do not account for wire delay. The contribution of these delays can be rather significant, especially for deep submicron CMOS nodes [25], i.e. for the cases we are mostly interested. Moreover, in the first two steps there are few places with a large fan-out on the critical path, which would require extra stages for an optimal design.

(ii) Our area estimates are based on the minimum-width gates. The optimization of delay will certainly require some gates to be larger than the minimum size and hence the area will be somewhat larger.

(iii) The circuitry can be optimized to reduce delay and possibly area by using faster circuit families. For example, static XOR gate is about 30% slower that the fastest implementation [26]. Moreover, the latency may be further reduced by balancing the delay of each stage [24] in the circuit.

The first issue is probably the most important and our estimates are likely to be on the optimistic side. Nevertheless, even such crude analysis might be enough to estimate performance overheads of ECC decoders in the prospective memory technologies with high defect rates.

ACKNOWLEDGMENT

Useful discussions with R. Koetter, K. K. Likharev, and T. Zhang are gratefully acknowledged.

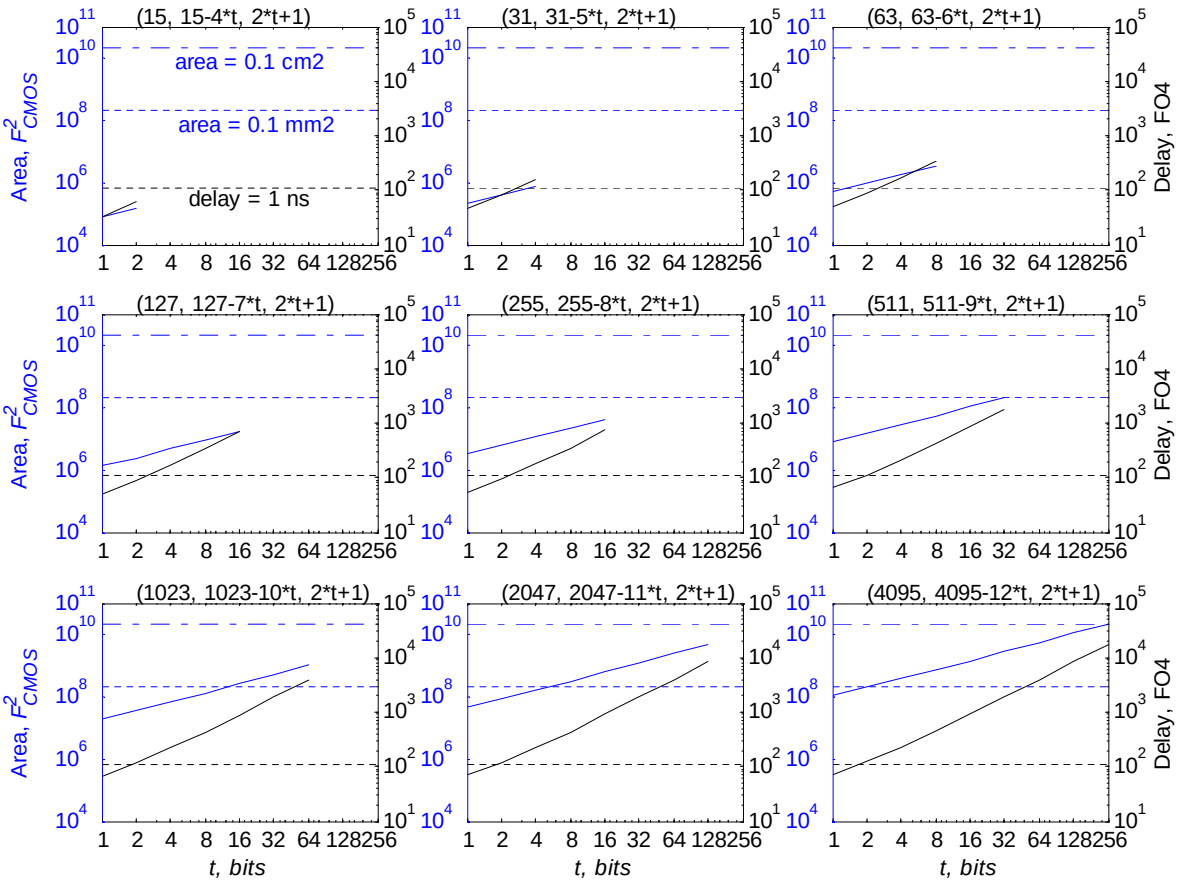

Fig. 4. The area and delay as a functions of the maximum number of errors t the BCH code can correct for several values of n. For convenience, horizontal lines show the delay of 1 ns, and the areas of 0.1 cm ^{2} and 0.1mm^2, for the 22 nm ITRS technology [22] nodes.

REFERENCES

[1] J. M. Tour, *Molecular Electronics: Commercial Insights, Chemistry, Devices, Architecture and Programming*. Singapore; River Edge, NJ: World Scientific, 2003.

[2] A. Chen, *et al.*, "Non-volatile resistive switching for advanced memory applications," *IEDM Tech. Digest*, p. 31.3, 2005.

[3] C. J. Amsinck, *et al.*, "Scaling constraints in nanoelectronic random-access memories," *Nanotechnology*, vol. 16, no. 10, pp. 2251–2260, 2005.

[4] C. Li, *et al.*, "Fabrication approach for molecular memory arrays," *App. Phys. Lett.*, vol. 82, no. 4, pp. 645–647, 2003.

[5] K. K. Likharev and D. B. Strukov, "CMOL: Devices, circuits, and architectures," in *Introducing Molecular Electronics*, G. Cuniberti, G. Fagas, and K. Richter, Eds. Berlin: Springer, 2005, pp. 447–478.

[6] D. B. Strukov and K. K. Likharev, "Prospects for terabit-scale nanoelectronic memories," *Nanotechnology*, vol. 16, pp. 137–148, 2005.

[7] ——, "Defect tolerant architectures for nanoelectronic crossbar memories," *J. Nanosci. Nanotechnol.*, 2006, in press.

[8] Y. Chen, *et al.*, "Nanoscale molecular-switch crossbar circuits," *Nanotechnology*, vol. 14, no. 4, pp. 462–468, 2003.

[9] W. Wu, *et al.*, "One-kilobit cross-bar molecular memory circuits at 30-nm half-pitch fabricated by nanoimprint lithography," *App. Phys. A-Mater. Sci. Process.*, vol. 80, no. 6, pp. 1173–1178, 2005.

[10] C. H. Stapper and H. S. Lee, "Synergistic fault-tolerance for memory chips," *IEEE Trans. Comput.*, vol. 41, no. 9, pp. 1078–1087, 1992.

[11] A. DeHon, *et al.*, "Nonphotolithographic nanoscale memory density prospects," *IEEE Trans. Nanotechnol.*, vol. 4, no. 2, pp. 215–228, 2005.

[12] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE Trans. VLSI*, vol. 9, no. 5, pp. 641–655, 2001.

[13] H. Lee, "High-speed VLSI architecture for parallel Reed-Solomon decoder," *IEEE Trans. VLSI*, vol. 11, no. 2, pp. 288–294, 2003.

[14] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge: Cambridge University Press, 2003.

[15] T. K. Matsushima, T. Matsushima, and S. Hirasawa, "Parallel encoder and decoder architecture for cyclic codes," *IEICE Trans. on Fundamentals*, vol. E79A, no. 9, pp. 1313–1323, 1996.

[16] M. A. Neifeld and J. D. Hayes, "Error-correction schemes for volume optical memories," *Applied Optics*, vol. 34, no. 35, pp. 8183–8191, 1995.

[17] C. Paar, *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*, Univ. of Essen, Essen, Germany, 1995, PhD thesis, Inst. for Experimental Math.

[18] C. Paar, P. Fleischmann, and P. Soria-Rodriguez, "Fast arithmetic for public-key algorithms in Galois fields with composite exponents," *IEEE Trans. on Comput.*, vol. 48, no. 10, pp. 1025–1034, 1999.

[19] A. DeHon and H. Naeimi, "Seven strategies for tolerating highly defective fabrication," *IEEE Des. Test Comput.*, vol. 22, no. 4, pp. 306–315, 2005.

[20] G. Snider, P. Kuekes, and R. S. Williams, "CMOS-like logic in defective, nanoscale crossbars," *Nanotechnology*, vol. 15, no. 8, pp. 881–891, 2004.

[21] E. D. Mastrovito, *VLSI architectures for computation in Galois fields*, Linkoping University, Linkoping, Sweden, 1995, PhD thesis.

[22] *International Technology Roadmap for Semiconductors. 2005 Edition*, available online at http://public.itrs.net/.

[23] M. A. Neifeld and J. D. Hayes, *Error-Correction Schemes for Volume Optical Memories*, 1995, vol. 34, no. 35.

[24] I. E. Sutherland, R. F. Sproull, and D. Harris, *Logical effort: Designing fast CMOS circuits*. San Francisco, CA; London: Morgan Kaufmann Publishers, 1999.

[25] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, 2001.

[26] H. T. Bui, *et al.*, "New 4-transistor XOR and XNOR designs," in *AP-ASIC-2000*, 2000, pp. 25–28.