A Defect/Error-Tolerant Nanosystem Architecture for DSP

WEIGUO TANG and LEI WANG
University of Connecticut
and
FABRIZIO LOMBARDI
Northeastern University

18

Emerging technologies such as silicon NanoWires (NW) and Carbon NanoTubes (CNT) have shown great potential for building the next generation of computing systems in the nano ranges. However, the excessive number of defects originating from bottom-up fabrication (such as a self-assembly process) poses a pressing challenge for achieving scalable system integration. This article proposes a new nanosystem architecture that employs nanowire crossbars for Digital Signal Processing (DSP) applications. Distributed arithmetic is utilized such that complex signal processing computation can be mapped into regular memory operations, thus making this architecture well suited for implementation by nanowire crossbars. Furthermore, the inherent features of DSP-type computation provide new insights to remedy errors (as logic/computational manifestation of defects). A new defect/error-tolerant technique that exploits algorithmic error compensation is proposed; at system level different trade-offs between correctness in output and performance are established while retaining low overhead in its implementation. As an instance of its application, the proposed approach has been utilized to a generic DSP nanosystem performing frequency-selective filtering. Simulation results show that the proposed nanoDSP introduces only a minor performance degradation under high defect rates and at a range of operational conditions. The proposed technique also features good scalability and viability for various DSP applications.

Categories and Subject Descriptors: B.8.0 [**Performance and Reliability**]: General

General Terms: Algorithms, Design, Reliability

Additional Key Words and Phrases: Distributed arithmetic, algorithmic error compensation, DSP nanosystem, inner product

Some preliminary results of this work were reported in the *International Symposium on Nanoscale Architectures (NANOARCH)* 2008 [Tang and Wang 2008].
This research was supported by the NSF grant CCF-0621947 and the University of Connecticut Faculty Research Grant 446751.
Authors' addresses: W. Tang, L. Wang, Department of Electrical and Computer Engineering, University of Connecticut, 371 Fairfield Way, U-2157, Storrs, CT 06269-2157; email: {weiguo.tang, leiwang}@engr.uconn.edu; F. Lombardi, Department of Electrical and Computer Engineering, Northeastern University, 309 Dana Research Building, Boston, MA 02115; email: lombardi@ece.neu.edu.

DOI 10.1145/1629091.1629094 http://doi.acm.org/10.1145/1629091.1629094

ACM Reference Format:
Tang, W., Wang, L., and Lombardi, F. 2009. A defect/error-tolerant nanosystem architecture for DSP. ACM J. Emerg. Technol. Comput. Syst. 5, 4, Article 18 (November 2009), 22 pages.
DOI = 10.1145/1629091.1629094 http://doi.acm.org/10.1145/1629091.1629094

1. INTRODUCTION

With lithography-based semiconductor technology quickly approaching the physical limits, emerging nanoelectronics such as silicon NanoWires (NW) and Carbon NanoTubes (CNT) have shown great potential for meeting the challenges of the Technology Roadmap beyond CMOS. Several crossbar-based nanosystems [Dehon and Wilson 2004; Stan et al. 2003; Wang et al. 2004; Dehon 2005; Goldstein and Budiu 2001; Ma et al. 2005] have been reported. It has been demonstrated that these repetitive nanostructures are suitable for implementing array-type logic, such as digital memory and programmable circuits. However, a large number of defects are likely to occur in bottom-up fabrication such as self-assembly. In emerging technologies, defects can be several orders of magnitude higher than CMOS. Thus, defect tolerance has emerged as a critical issue to sustain functional integration at nano scale.

Many techniques have been proposed to improve defect tolerance in nanosystems; these range from reconfiguration, spatial or temporal redundancy, Error Checking Codes (ECC), encoded computing, or a combination of several of them depending on the context and application. Postfabrication reconfiguration employs mapping to avoid defective devices and adopts spare (defect-free) devices as functional units. Usually, a test/diagnosis process at per-chip basis is required for a successful replacement of defective components. Reconfiguration can be implemented by mapping [Heath et al. 1998; Koren et al. 1994; Tahoori 2005] to identify the defective devices, or a reconfigurable system can detect defects autonomously without external intervention [Manger 2000; Macias 1999; Macias and Anthanas 2007]. Redundancy-based techniques [Jeffery and Figueiredo 2006; Lee et al. 2004] utilize the inherent redundancy in nanoscale integration to operate with defective devices and thus achieve defect tolerance. The classical approach of NAND multiplexing originates from von Neumann [1956] in which a large integer factor overhead is involved (i.e., a degree of redundancy $R > 100$). Different from reconfiguration that targets permanent defects, NAND multiplexing is also effective in dealing with transient faults. A further study in Han and Jonker [2002] improves the von Neumann's NAND multiplexing method with a relative lower degree of redundancy ($R < 100$). The joint application of both reconfiguration and NAND multiplexing in Han and Jonker [2003] exploits the advantage of both techniques to further reduce redundancy by a factor of 10. In addition to spatial redundancy, methods relying on temporal redundancy (e.g., rollback [Mizan 2007; Naeimi and DeHon 2008]) are effective when the error rate is low. ECC is commonly employed to provide fault tolerance for nanoscale memories and interconnects [Kuekes et al. 2005; Strukov and Likharev 2007; Sun et al. 2008]. While significantly difficult, computing on encoded data has also been investigated [Nair and Abraham 1990;

Spielman 1996; Rachlin and Savage 2008] for defect/fault tolerance. In general, the redundancy of ECC is less than the other two previously described approaches. This is, however, at the expense of complex encoding/decoding circuits. Most of these techniques aim at achieving absolute correctness for general-purpose computing. In Lee et al. [2005], a fault-oriented test method was proposed to identify a set of faults that can be ignored during manufacturing test as long as the system error rate requirement is not violated. This method, however, is not directed to nanosystem design.

It is widely acknowledged that emerging technologies are unlikely to compete with CMOS for general-purpose computing in the near future. Thus, in addition to memory which is the likely application targeted by emerging technologies, new application domains must be identified as the starting point to pursue an investigation. In this article, we propose to employ nanowire crossbars for digital signal processing by exploiting the property that DSP applications typically do not require absolute correctness, and features such as algorithm-level error resilience, parallel/distributed computation, and ease of programmability can be naturally extended for defect and error tolerance at the nano range. A DSP nanosystem architecture (referred to as *nanoDSP*) based on Distributed Arithmetic (DA) [Peled and Liu 1974] is proposed; it matches well to the array-based computing in nanowire crossbars. Defect tolerance is achieved by exploiting algorithmic error compensation via a low-complexity Error Correction Table (ECT). The underlying principle is to identify the patterns of defect-induced errors (as manifestation of physical phenomena) and assign an error correction capability according to the algorithmic significance in determining system performance. This is shown an effective and efficient method to achieve defect tolerance for DSP at nano scale. Compared with existing defect-tolerant techniques, the proposed technique enables better scalability with low overhead and complexity.

Some preliminary results were reported in Tang and Wang [2008]. In this article, we extend our past work by making the following contributions. We propose an enhanced ECT by jointly considering the magnitude and the occurrence frequency of hardware-induced errors. This is an effective approach by not only allowing more efficient allocation of ECT resources but also ensuring the scalability of the proposed approach under a range of potential operational scenarios. Also, we develop a design methodology to deal with long memory words, which are more vulnerable to defects and thus errors. By dividing long memory words into several subwords and assigning an algorithmically weighted error protection, this design optimization significantly reduces the hardware overhead. We also consider the defects in the address multiplexer and develop a configuration process to alleviate the impact of the defective address multiplexer on the performance of the nanosystem. Finally, we conduct extensive simulations to evaluate the performance of the proposed approach.

The rest of the article is organized as follows. In Section 2, we review the nanowire technology and examine the defects occurring in nanowire crossbars. In Section 3, we present the DA-based nanoDSP exploiting algorithmic error compensation for tolerance of defects in nanowire crossbars. We also perform a statistical analysis on the proposed approach. In Section 4, we discuss the

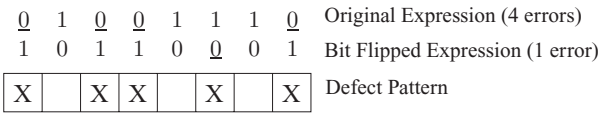

Fig. 1. An example of reducing defect-induced errors by conditional bit flipping.

design methodology for long memory words. The evaluation of the proposed nanoDSP is provided in Section 5.

2. REVIEW OF NANOWIRE CROSSBARS

Nanowire crossbars are considered as promising building blocks for designing next-generation nanocomputing systems. Nanowire crossbar circuits can be implemented using a number of vertical and horizontal metal wires on two parallel planes. At each crosspoint of the two nanowires, there is a Pt/rotaxane/Ti junction, which can be configured to a low resistance (between 10^6 and $5 \times 10^8\Omega$) or a high resistance ($\geq 4 \times 10^9\Omega$) to represent logic "0" and "1", respectively. All configuration operations are performed upon the crosspoints, which can be implemented by resistors, diodes, or Field-Effect Transistors (FET). Thus, not only logic functions but also memory states can be realized.

Various defects may occur during bottom-up (self-assembly) fabrication. It is projected that defect rates in the range of 5%–10% will pose a significant challenge for constructing complex logic functions and memory systems. Most defects in nanowire crossbars can be placed into three broad categories: (1) *crosspoint stuck-at-open*, caused by missing switches at crosspoints; (2) *crosspoint stuck-at-closed*, in which two orthogonal nanowires are shorted together at a crosspoint; and (3) *nanowire open*, indicating broken nanowires. Stuck-at-closed and nanowire open defects can be considered as nanowire defects. These two types of defects are equivalent as both are detrimental to all crosspoints located on the two affected orthogonal nanowires. Hence, the involved nanowires along with the crosspoints should be mapped onto a reconfiguration step. A stuck-at-open defect makes an output stuck at "1"; hence it will not cause an error if the expected read-out value is a logic "1". Therefore, to store as many "1"s as possible in the defective crosspoints may reduce bit errors.

Assume that there are $2N_d + 1$ stuck-at-open defects in a memory word. We can flip (change) all bits if the data has less than $N_d + 1$ of "1"s in the defective crosspoints. This will keep the bit-wise errors (caused by stuck-at-open defects) to be less than $N_d + 1$. Consider the example illustrated in Figure 1. We assume five stuck-at-open crosspoints in an eight-bit word. The direct representation has just one 1-value bit using the defective crosspoints and this results in four bit errors. By flipping all the bits, four 1-value bits will use defective crosspoints, thereby reducing the number of bit errors to just one. This technique is referred to as *Conditional Bit Flipping* (CBF), and has been employed in some existing work [Sun et al. 2008] to reduce the complexity of Error Checking Codes (ECC) for memory nanosystems. In general, the utilization of only CBF is not sufficient to satisfy all defect and error tolerance requirements of nanowire crossbars.

3. DA-BASED NANODSP WITH DEFECT/ERROR TOLERANCE

In this section, we propose to employ Distributed Arithmetic (DA) for the design of nanoDSP systems in nanowire crossbars. The proposed nanoDSP is a memory-based architecture which turns complex DSP-type computation into regular memory operations. By exploiting the inherent features of signal processing, low-complexity yet effective defect/error tolerance can be achieved.

3.1 Distributed Arithmetic

One of the most frequently encountered forms of computation in DSP is the inner product of a pair of vectors, which can be computed directly as

$$y = \sum_{i=0}^{L-1} a_i x_i, \tag{1}$$

where $\{a_i\}$ may be a constant vector (e.g., the coefficients of an FIR filter) and $\{x_i\}$ are the input data.

Distributed Arithmetic (DA) performs a bit-serial operation that computes the preceding inner product in a distributed and parallel fashion. Consider the input x_i in 2's-complement binary format expressed as

$$x_i = -b_{i(N-1)}2^{N-1} + \sum_{k=0}^{N-2} b_{ik}2^k, \tag{2}$$

where $b_{ik} \in \{0, 1\}$ is the k^{th} bit of x_i and the Most Significant Bit (MSB) $b_{i(N-1)}$ is the sign bit. Substituting x_i from (2) into (1), we can recast the computation of the inner product as

$$y = -\sum_{i=0}^{L-1} a_i b_{i(N-1)}2^{N-1} + \sum_{k=0}^{N-2} \left[\sum_{i=0}^{L-1} a_i b_{ik}\right] 2^k. \tag{3}$$

If we define a new term as

$$A_m = \left[\sum_{i=0}^{L-1} a_i b_{ik}\right], \tag{4}$$

then for different values of input bits $b_{0k}, b_{1k}, \ldots, b_{(L-1)k}$, A_m will have only 2^L possible values taken from the different combinations of $\{a_i\}$ (e.g, $A_1 = a_0$, $A_2 = a_0 + a_1$, etc.). These values can be precomputed and stored in a memory of size 2^L. The inner product (3) can be calculated by accessing this memory N times using input bits $b_{0k}, b_{1k}, \ldots, b_{(L-1)k}$ as the address, and accumulating the corresponding memory data to generate the final result. Since this computation is inherently distributed and performed in parallel, the disadvantage of bit-serial operations can be almost completely hidden.

Figure 2 shows a generic DSP module implemented in DA. All the A_m's are stored in a memory block with 2^L entries. The input bits $b_{0k}, b_{1k}, \ldots, b_{(L-1)k}$ are used as the address to map the corresponding memory entry. For example, if $\{b_{0k}b_{1k} \cdots b_{(L-1)k}\} = 1010 \cdots 0$, then the memory entry storing $a_0 + a_2$ is accessed. The weight of the accessed data is adjusted (e.g., by shifting it left) and

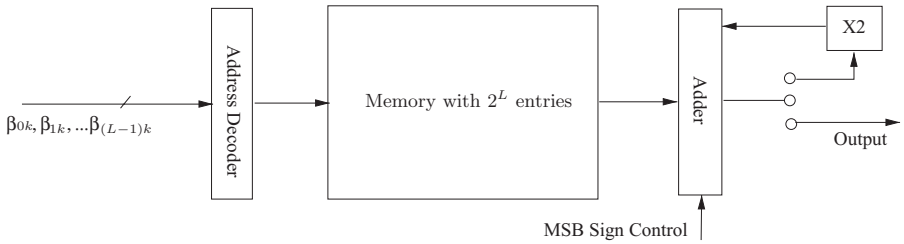

Fig. 2. Diagram of a generic computing module with distributed arithmetic.

the result is added to an intermediate value. This procedure starts from the MSBs of $x_0, x_1, \ldots x_{L-1}$ and continues until the LSBs are processed. The result corresponding to the sign bits $b_{0(N-1)}, b_{1(N-1)}, \ldots, b_{(L-1)(N-1)}$ is subtracted from the final result.

3.2 Algorithmic Error Compensation

A DA-based DSP architecture is well suited for nanowire crossbars, as they are efficient in implementing array-type logic such as memory. Furthermore, signal processing provides new means to manage memory bit errors caused by defects in nanowire crossbars. In this subsection, we will present a new defect-tolerant technique, referred to as *Algorithmic Error Compensation* (AEC), for DA-based nanoDSP systems.

In most DSP systems, bits contribute differently to the algorithm-level performance of the application. Terms such as the Most Significant Bit (MSB) and the Least Significant Bit (LSB) are self-explanatory. Thus, error tolerance in DA-based nanoDSP must be assigned to the data in the nanowire crossbar memory according to their algorithmic significance. This is intuitive because DSP applications typically do not require absolute correctness in the computing process, that is, the hardware-induced errors can be regarded as part of the signaling noise inherent in the signal processing tasks. When hardware-induced errors are small in magnitude or within an acceptable range (e.g., mainly from LSBs), a minor performance degradation (e.g., reduction in Signal-to-Noise Ratio (SNR) or increase in Bit Error Rate (BER)) may be introduced; however, it will not likely cause fatal system failures. By exploiting these properties, we propose to use Algorithmic Error Compensation (AEC) that leverages the algorithmic capability of DSP applications to address the physical complexity and the possible defects in emerging technologies.

Specifically, the proposed AEC maintains an Error Correction Table (ECT) that stores error patterns indicating the positions of bit errors in a memory data. For example, the error pattern of an 8-bit memory data with errors in the 1^{st} (i.e., the LSB) and 4^{th} bits is expressed as 00001001. Thus, error correction can be accomplished by an XOR operation between the data and its error pattern in the ECT. The size of the ECT is set by the word length of the memory data and the number of bit errors that it intends to correct. As discussed in the following sections, these factors are directly related to the algorithmic requirements of

a DSP application. With no loss of generality and correctness, we will first consider a generic case for the purpose of illustration.

Assume that the word length of the memory data is N bits and errors as manifestation of defects are uniformly distributed. Then, the number of error patterns indicating just one bit error is N, and the number of error patterns for exactly two bit errors is $\binom{N}{2}$. In general, the number of error patterns for d_e bit errors is $\binom{N}{d_e}$. Therefore, the total number of possible error patterns for correcting up to d_e bit errors in a nanoware crossbar memory is given by

$$L_{ep}(d_e, N) = \sum_{i=0}^{d_e} \binom{N}{i}. \tag{5}$$

For convenience, let d_e denote the error correction capability of the ECT of size $L_{ep}(d_e, N)$. From (5), the size of the ECT is a function of the memory word length N and the error correction capability d_e but it is independent of the number of memory entries (e.g., L in (4)). This is because all memory entries share the same set of error patterns. Thus, ECT is scalable with memory size (in terms of the number of entries) as it only induces a fixed overhead.

Consider a signal processing task requiring 16-bit precision. The DA-based nano-DSP needs $64K$ (2^{16}) memory words where each word is 16 bits. From (5), to achieve the error correction capability at $d_e = 2$, only 137 error patterns need to be maintained in the ECT. Due to its small size, ECT can be implemented in a reliable and more stable technology such as CMOS; or it can be implemented in nanowire crossbars using conventional defect-tolerant techniques such as reconfiguration- or redundancy-based techniques. In either case, the hardware overhead is manageable.

The error pattern of a memory entry can be determined prior to normal computation because data is precomputed in DA-based nanoDSP (see Section 3.1). The error pattern assignment must consider algorithm-level implications, such that the ECT can provide effective defect tolerance for DSP-type computation. In particular, we always map defective bits starting from the MSB to the LSB to form the error patterns. If the number of bit errors in a memory entry exceeds the error correction capability d_e, then we choose the first d_e bit errors (from the MSB) as the error pattern. For example, when the actual error pattern of a memory entry is 01001001 (containing three bit errors) and the error correction capability is $d_e = 2$, then we will choose the first (higher order) two bit errors to form the error pattern of 01001000 for this memory entry. Therefore, errors may occur in the bits of less significance, but this will only introduce minor degradation in algorithmic performance (refer to Section 5 for more detail). Although not all hardware-induced errors can be corrected, the impact of nanowire defects is greatly contained from an algorithmic viewpoint, because only the LSBs can be corrupted.

Each defective memory entry is associated with an error pattern in the ECT. An address multiplexer can be utilized to link the main memory and the ECT. The trade-offs related to the multiplexer design will be discussed in Section 3.4. For architectural design, key parameters such as memory size (i.e., 2^L), word length (N), and error protection capability (d_e) are related to system

performance requirements (e.g., SNR or BER). The size of the main memory is determined by the length of the inner product vectors; this is a function of the precision of the DSP applications. The word length and error protection capability must be selected based on performance requirements and defect rates. In general, a higher SNR will require a longer word length and a larger protection capability under the same defect rate. We can also increase the word length to improve the SNR without using a high protection capability if the defects rate is low. The gain for increasing the protection capability is upper bounded by the word length N. When the residual error rate is low, the effectiveness of increasing the protection capability is trivial. A more detailed analysis is given in Section 3.5.

3.3 Enhanced ECT

The AEC technique discussed in the previous section constructs the ECT based on a fixed d_e. This approach is effective when the defect rate is relatively low and defects are uniformly distributed. In such cases, the hardware-induced errors in a memory entry will occur only in few bits and thus they can be corrected (at least partially) by the ECT at a relatively small d_e. However, for nonuniformly distributed (e.g., clustered) defects or when the defect rate is rather high, errors may occur in many bits of a memory entry, thus posing a significant concern. This requires a large d_e which in turn increases the overhead of ECT exponentially.

To address this problem, it is desirable to utilize a so-called enhanced ECT that can reduce as many errors as possible. From this perspective, we may need to extend the error correction capability to more than d_e bits. However, to avoid a large size ECT, selection of the error patterns must be performed wisely. This can be done by considering the contribution of each error pattern; for example, having both error patterns 11110000 and 11110001 in the ECT is not a good choice, because the second pattern is better than the first one only with respect to the LSB. Hence, the contribution of the second pattern is almost negligible if the first pattern is already in the ECT.

Under such an observation, we need a metric for measuring the contribution of an error pattern to the reliable performance of nanoDSP systems. The contribution of an error pattern can be quantified by the additional bit errors that can be corrected (or reduced) by storing this error pattern in the ECT. For example, assume that the ECT already has two error patterns 10000000 and 10001000, and we want to add a new error pattern 10001100. This new error pattern can be used to correct the error 10001100. If we don't have any other error patterns in the ECT, this error pattern can also be used to reduce errors $100011XX$, where "X" (i.e., don't care) can be either "0" or "1", that is, by applying the existing two error patterns this will result in larger residual errors. However, errors such as 10000000 and 10001000 can be corrected by the existing two error patterns. Thus, the net contribution of the new error pattern 10001100 is to correct the error in the third bit from the LSB. This error is defined as the *intrinsic correctable error* of this error pattern. Note that in general, the intrinsic correctable error of an error pattern may have multiple

bit errors (i.e., the 1-value bits). We can calculate the contribution of an error pattern as

$$\eta = p_{cr} w, \tag{6}$$

where p_{cr} is the probability of the errors that can be corrected or reduced by the error pattern, w is the weight of the intrinsic correctable error of the error pattern given by

$$w = \sum_{i=1}^{k} 2^{q_i}, \tag{7}$$

k is the total number of bit errors in the intrinsic correctable error, and q_i is the index of the i^{th} bit error with MSB being the $N-1$ and LSB being 0 for a N-bit error pattern.

When calculating p_{cr}, all the errors that can be either corrected or reduced by introducing a new error pattern must be considered. Consider again the previous example, these errors are in the form of $100011XX$ for the error pattern 10001100. If defects are uniformly distributed, p_{cr} can be derived as

$$p_{cr} = p^m (1-p)^{N-m-k_{null}}, \tag{8}$$

where p is the defect rate, m is the number of "1"s in the error pattern, and k_{null} is the number of "0"s following the rightmost "1" in the error pattern (e.g., $k_{null} = 2$ for error pattern 10001100). For other defect distributions, the corresponding p_{cr} must be established to assess the contribution of an error pattern. In general, the number of bit errors will increase significantly as the defect rate p increases.

From (6)–(8), the enhanced ECT takes into account both the magnitude and the occurrence frequency of hardware-induced errors, while the basic ECT in Section 3.2 only considers the magnitude of the errors. As the performance of DSP nanosystems is affected by not only the magnitude but also the occurrence frequency of hardware-induced errors, then the enhanced ECT is in general a better solution. This is evident from the discussion in the following sections.

By employing the enhanced ECT, we can select error patterns to maximally reduce the algorithmic impact of hardware-induced errors at no additional ECT overhead (i.e., the size of the enhanced ECT can be the same as the basic ECT). Furthermore, the enhanced ECT can be adjusted by adding or removing error patterns (based on their contributions) to achieve different error protection levels. The basic ECT lacks this flexibility because the size of the ECT increases exponentially with d_e (e.g., $L_{ep}(2, 16) = 137$ and $L_{ep}(3, 16) = 697$), leading to either an under- or an overprotection to errors. While the size of the enhanced ECT is no longer directly related to d_e, we can still map errors in a memory entry to the error patterns in the ECT. This is accomplished by starting from the MSB and continuing until the closest error pattern is located. This follows a similar procedure as the one discussed in Section 3.2.

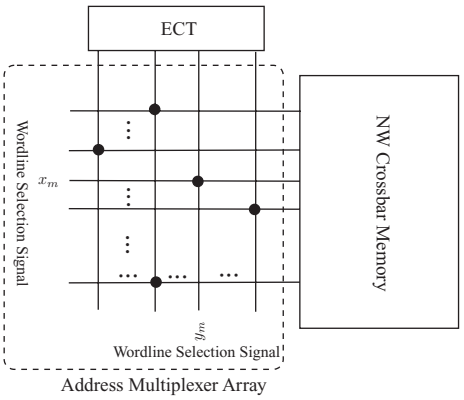

Fig. 3. Address multiplexer array in NW crossbars.

3.4 Design Options for the Address Multiplexer

Our previous work [Tang and Wang 2008] has discussed a possible implementation of the address multiplexer; in this implementation, a programmable crossbar array connects the wordlines of the main memory to the corresponding ECT entries as illustrated Figure 3. The address multiplexer design leads to some unique trade-offs that need to be fully understood. Specifically, the address multiplexer in Figure 3 appears to be larger than the main memory; however, it requires a much smaller number of crosspoints to be programmed (i.e., at most one crosspoint per row), and thus is less vulnerable to defect-induced errors. Since reliability is one of the dominant challenges in nanowire crossbar architectures, this address multiplexer implementation may still be a viable option if it can provide reliable mapping between the ECT and main memory.

Reliable mapping is a relative metric because a defect-free nanowire crossbar is not possible. As the address multiplexer also sees a large number of defects, then mapping the desired error patterns (as derived in the previous sections) may not always be achieved. So, we investigate a mapping procedure to alleviate the impact of defects in the address multiplexer. We will show that it is possible to achieve good algorithmic performance even when some mappings between the ECT and main memory are defective. This is in line with the main theme of our article, namely DSP applications do not require absolute hardware correctness.

In this article, the address multiplexer mapping consists of two steps: error pattern assessment and assignment. The key observation in the error pattern assessment is that error patterns contribute differently to the performance degradation of a DSP system. Thus, even if a perfect mapping between the ECT and main memory may not be possible due to defects in the address multiplier, we can still find a mapping configuration that has the least degradation in performance. For each error pattern in the main memory, we can assess its significance using the enhanced ECT method as discussed in Section 3.3.

During the process of error pattern assignment, we configure the address multiplexer to connect the ECT with the main memory by using the order of significance in error patterns. As we assume stuck-at-closed defects are mapped out through a reconfiguration step (otherwise these defects are detrimental because all crosspoints on the affected nanowires are unusable), then a stuck-at-open defect will not cause a mapping error if it occurs in an unused crosspoint. Thus, we assign error patterns to the address multiplexer columns to reduce the impact of defect-induced mapping errors. Theoretically this minimization problem is NP hard; so we adopt a greedy-based approach. The most significant error pattern will be assigned first to a column for a mapping configuration with no (whenever possible) or minimal defect-induced mapping errors. As this process continues, mapping less significant error patterns will more likely be affected by the defects in the address multiplexer; however, the impact of mapping errors will be reduced as much as possible. In some cases, we may need to replace an error pattern with a similar one that can be mapped successfully. For example, if the mapping of the error pattern 10000001 for one entry of the main memory is impossible due to the defect at the corresponding crosspoint, we may instead use the error pattern 10000000 for this entry if its mapping is defect free. This leads to a partial error correction in which only errors in the LSBs are not corrected. In the worst case, defects in the address multiplexer may make impossible to map some error patterns with less significance. As most of the mapping errors occur in the error patterns of less significance (e.g., in the LSBs), the proposed mapping procedure will not incur in a large degradation in algorithmic performance (less than 1dB SNR loss under a defect rate of 20%, as shown in Section 5).

Another option for the design of the address multiplexer is to encode the addresses to reduce the multiplexer size. For example, with $d_e = 2$ and $N = 16$, the selection signals of the multiplexer are 8 bits (as compared with 137 bits in the previous work), in which multiple crosspoints have to be configured for each main memory entry). Conventional redundancy-based techniques may be applied to handle defects in such an implementation. However, a trade-off between complexity and reliability must be assessed to balance defect tolerance with design overhead. Since the ECT may be implemented in a reliable and more stable technology (such as CMOS), the overhead of the multiplexer is still likely to be smaller than the ECT (refer to Section 5).

3.5 Performance Analysis

The proposed AEC shows several advantages compared with existing techniques. In particular, it allows flexibility to select error patterns that contribute more to the algorithmic performance of DSP nanosystems. For example, we can correct hardware-induced errors on the MSBs if the number of bit errors exceeds the error correction capability d_e. This is a very effective and efficient method to achieve defect/error tolerance in DSP nanosystems. Conventional ECC in general provides equal protection to all memory bits. This may be more suitable for general-purpose computing, but it also introduces a large complexity and overhead in error detection and correction. Although some ECC schemes can

be implemented with a weighted protection for the different bits, the complexity will significantly increase. In comparison, the ECT provides good coverage on hardware-induced errors with only a small overhead. To correct up to 2 bit errors, the size of the ECT (in terms of number of memory bits) only accounts for 0.2% of the 64k memory. In addition, error correction only involves a simple bit-wise XOR operation between the corrupted data and the error patterns. Finally, the overhead of the proposed approach is relatively unrelated to the domain complexity of DSP applications. The size of the ECT is only a function of the word length N and the error correction capability d_e. For complex nanoDSP systems requiring a larger number of memory entries, the overhead due to the ECT is fixed because the ECT is shared by all memory entries. This indicates that the proposed approach features good scalability for DSP applications.

Next, a statistical analysis is presented to provide a guideline for assessing the performance of the proposed AEC assuming defect-free address multiplexer configuration. The Mean Square Error (MSE) after error correction is utilized for comparison. We first consider the case in which no error protection is applied. Since the error patterns describe the bit difference (not exactly the numerical difference) with respect to the error-free data, then the MSE can be calculated as

$$\sigma^2 = \sum_{i=1}^{N} 2^{2(i-1)} p = \frac{p(4^N - 1)}{3}, \tag{9}$$

where p is the defect rate and N is the memory word length.

For AEC with the basic ECT, if there are more than d_e bit errors, the first d_e bit errors (starting from the MSB) will be recovered, whereas the errors in the remaining $N - d_e$ bits will become the residual error. Thus, the MSE can be obtained as

$$\sigma^2_{BECT} = \sum_{k=1}^{N-de} 2^{2(k-1)} p \left[\sum_{l=de}^{N-k} \binom{N-k}{l} p^l (1-p)^{N-k-l} \right]. \tag{10}$$

Figure 4 shows the results of the AEC with the basic ECT under different defect rates, where the MSE is evaluated by the equivalent SNR, that is, the ratio between the desired signals and the defect-induced errors. An increase in the error protection capability d_e will significantly increase the SNR which indicates large reduction in the residual error. On average, about 15 dB improvement is observed for every increase in d_e.

For AEC with the enhanced ECT, it is difficult to derive the closed-form expression for the MSE. We instead evaluate this metric through simulation. Figure 5 shows the results of the equivalent SNR for three designs, in which the enhanced ECT has the same size as the basic ECT. As expected, the enhanced ECT achieves the best performance for error tolerance. For example, when $p = 10\%$, the enhanced ECT increases the SNR by more than 20dB over the basic ECT. This improvement is evident for various defect rates. Even at high defect rates, the residual errors after applying the ECT are still very small.

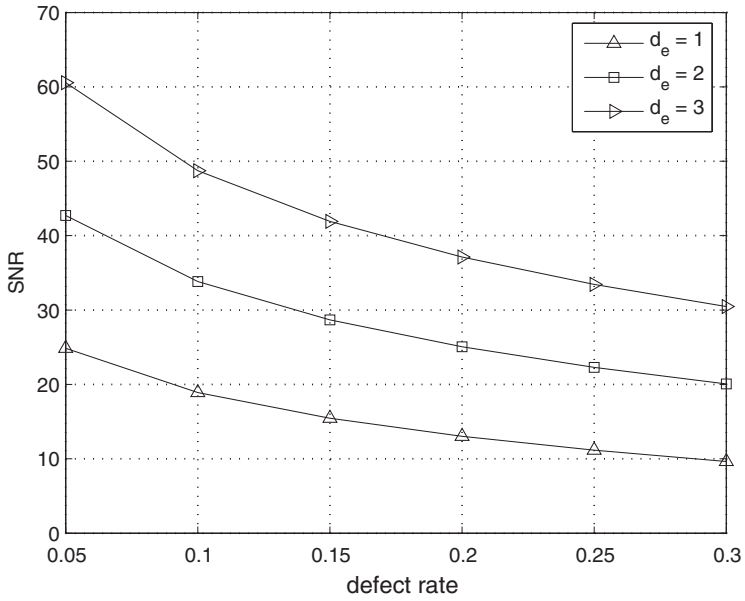

Fig. 4. The SNR of the proposed scheme with the basic ECT with $N = 16, d_e = 1, 2, 3$.

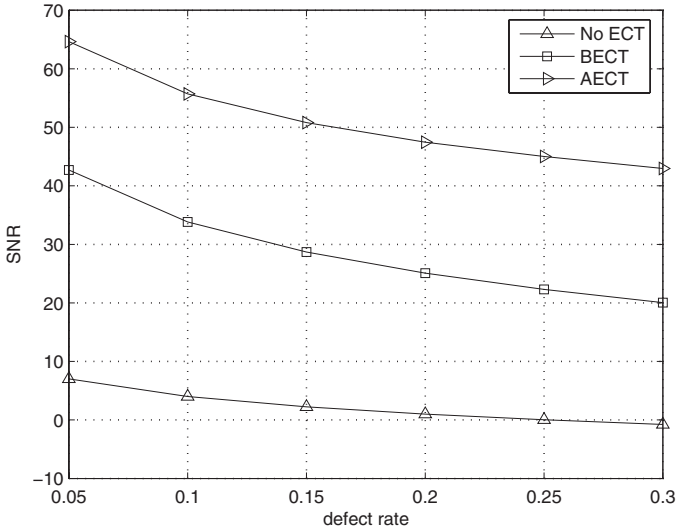

Fig. 5. Comparison of the SNR for three different designs with $N = 16, d_e = 2$.

4. DESIGN FOR LONG MEMORY WORDS

From (5), L_{ep} is determined by the word length N and the error correction capability d_e. Thus, the size of the ECT will grow quickly as N increases. This problem is made worse by the need to increase d_e for long words because they are more vulnerable to defects. In DSP applications, long words imply high

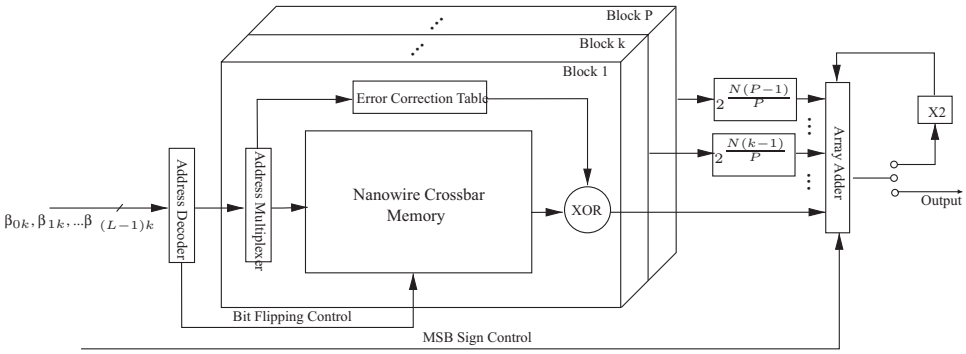

Fig. 6. The proposed DA-based nanoDSP module in which a long word is evenly divided into P subwords.

precision such that defect/error tolerance is even more critical. The ECT (even the enhanced ECT) will have to be large enough to achieve good error management, but this inevitably increases the complexity and overhead in DA-based nanoDSP systems.

To address this problem, we propose a design methodology that partitions each long memory word into several subwords and stores these subwords in smaller memory blocks. The subwords of the same word can be accessed by the same set of input bits; thus, we can combine them back to obtain the original data. In this implementation, the ECT only needs to protect the subwords. As $PL_{ep}(N/P, d_e) \ll L_{ep}(N, d_e)$ (where P denotes the number of subwords), the number of error patterns is reduced significantly leading to a much smaller overhead. The architecture of this design is shown in Figure 6, in which each long word is partitioned equally into P subwords. The P^{th} subword stores the MSBs of the original word. The data read from the k^{th} memory block is shifted left by $\frac{N(k-1)}{P}$ bits to generate the final output.

Error management of this architecture must consider the significance of subwords in determining the algorithmic performance. The subwords adjacent to the MSB are more important; thus, we can assign a higher protection level (i.e., increasing d_e for basic ECT) to those memory blocks storing MSBs and accordingly reduce the protection level for the other memory blocks. For the basic ECT as an example, we can divide a 32-bit word into two 16-bit subwords and assign $d_e = 3$ and $d_e = 1$ to the high-order and low-order subwords, respectively. The benefits of the enhanced ECT are evident in this scenario because many choices are available for tailoring the size of the ECT to the algorithmic significance of the different memory blocks. These trade-offs will be studied in the next section.

Algorithm 1 summarizes the algorithmically weighted design methodology. Given the number of subwords, the objective is to meet the MSE requirement for the desired system performance with a small ECT overhead. Without loss of generality, we initially assign the same d_e to all subwords. We then search for the high-order subwords and increase their error protection capability. This process iterates on a greedy basis until the performance requirement is met or exceeded. If the performance requirement is exceeded (i.e., the system is

Algorithm 1. Algorithmically Weighted Error Protection

Input:
N (Total word length)
P (Number of subwords)
σ_0 (MSE to meet performance requirements)
$\vec{d_{e0}}$ (Initial error protection capability, assuming equal for all subwords)
begin
  $\sigma = calculate\_MSE(P, N, \vec{d_{e0}})$
  while $\sigma > \sigma_0$ **do**
    %find the maximal index with minimal protection capability
    $i = max\_min(\vec{d_e})$
    $d_e[i] = d_e[i] + 1$
    $\mu = calculate\_MSE(P, N, \vec{d_e})$
  end
  while $\sigma < \sigma_0$ **do**
    %find the minimal index with maximal protection capability
    $i = min\_max(\vec{d_e})$
    $d_e[i] = d_e[i] - 1$
    $\sigma = calculate\_MSE(P, N, \vec{d_e})$
    if $\sigma > \sigma_0$**then**
      $d_e[i] = d_e[i] + 1$
      remove $d_e[i]$ from the $\vec{d_e}$
    end
    if $\vec{d_e}$ is empty
      Break
    end
  end
end

over protected), a reduction in the error protection capability of low-order subwords is utilized.

5. EVALUATION

In this section, we will study the performance of the proposed DA-based nanoDSP in a digital filter application. For ease of presentation, we have selected a lowpass Finite Impulse Response (FIR) filter to suppress the out-of-band interference and improve the quality of the desired signal with bandwidth $[0, 0.4\pi]$. An equiripple lowpass filter is implemented using the proposed technique with order of 11, passband 0.4π, stopband 0.6π, and stopband fading 30dB. Assume that the memory block and address multiplexer are designed with nanowire crossbars while the other logic functions and the ECT table are implemented in CMOS. Thus, most of the defect-induced errors will originate from the nanowire crossbar memory. We choose a range of defect rates and input Signal-to-Interference Ratios (SIR) and measure the Signal-to-Noise Ratios (SNR) at the output. The output noise contains two components: the residual

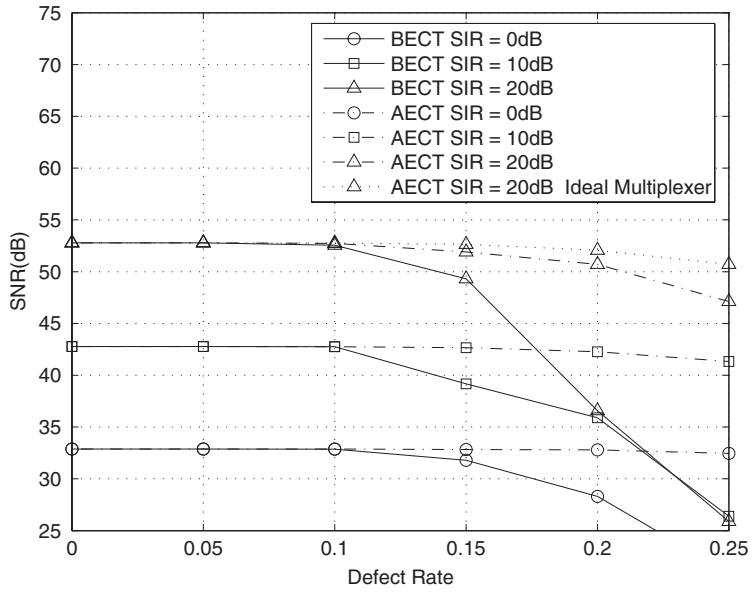

Fig. 7. Output SNR of a DA-based nanoDSP implementing a FIR filter with $N = 16, d_e = 2$.

signaling noise from the interference at the input and the defect-induced errors. Thus, a high defect tolerance will result in a smaller number of defect-induced errors and thus a higher SNR at the output. Defects are assumed uniformly distributed and the output SNR is obtained by averaging over 100 simulation runs with different input samples. The enhanced ECT and the basic ECT are referred to as AECT and BECT, respectively, in all figures. The size of the enhanced ECT is denoted by L_{AECT}.

Figure 7 shows the results of the basic ECT with $d_e = 2$ and the enhanced ECT with same size. The word lengths of both the input signal and the nanowire crossbar memory are 16 bits. When the basic ECT is employed, the nanoDSP module works well for defect rate up to 10% without deteriorating the quality of the filter output (e.g., no SNR degradation). If the requirement of the output SNR is no more than 30dB, the nanoDSP module can tolerate a defect rate up to 15% with a slightperformance loss of less than 1dB. Even when the defect rate increases up to 20%, the performance loss is about 4dB, which is still within an acceptable range for many signal processing applications. In the simulations, the size of the memory block is given by 4k words and the ECT only needs to store 137 error patterns. The proposed nanoDSP can perform reliable DSP-type computation at a small overhead of error management.

From Figure 7, the enhanced ECT achieves almost the same performance as the basic ECT when the defect rate is less than 10%. When the defect rate increases, the advantages of the enhanced ECT become more evident. The nanoDSP module with the enhanced ECT can tolerate up to a 20% defect rate with no performance loss when the input SIRs is less than 20dB. When the input SIR is high (e.g., 20dB), the output SNR will degrade by less than 1dB and about 2.5dB when the defect rate is 15% and 20%, respectively. Moreover,

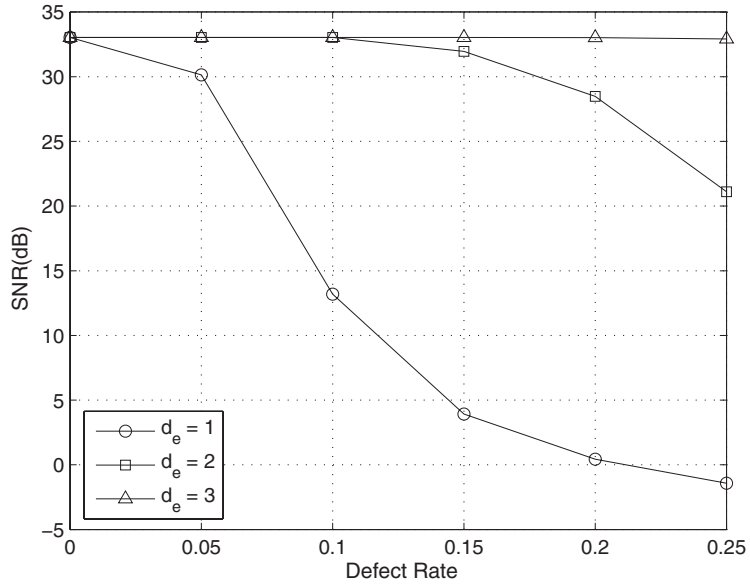

Fig. 8. Effects of different error correction capabilities on the output SNR with $N = 16$, $SIR = 0dB$.

the enhanced ECT can operate reliably for defect rate up to 25% with no performance loss when the SIR is lower than 10dB.

The reason for the better performance of the enhanced ECT versus the basic ECT at high defect rates can be explained as follows. When the defect rate is low, the hardware-induced errors are dominated by error patterns with few error bits. In this case, the basic ECT with a fixed d_e is sufficient to correct these errors. As the defect rate increases, we expect to see more bit errors beyond the capability d_e of the basic ECT. The enhanced ECT effectively deals with them by considering the contributions of the error patterns and selecting those with large impacts. This observation is consistent with the results of the MSE as discussed in Section 3.4.

All the simulation results assume the same defect rate for the address multiplexer. Applying the mapping procedure as discussed in Section 3.4, the defects in the address multiplexer lead to only minor performance degradation. For SIR being equal to 20dB as an example, there is almost no SNR loss due to the nonideal mapping in the address multiplexer when the defect rate is less than 15%. When the defect rate reaches 20%, we observed less than 1dB SNR loss.

Next, we consider the size of the ECT and its impact on performance. In the basic ECT, the size is mostly set by the error correction capability d_e. Thus, the selection of d_e has a direct implication on the design of DA-based nanoDSP systems. Figure 8 shows the results for different d_e's. When d_e is small (e.g., $d_e = 1$), the algorithmic performance degrades quickly with an increase of the defect rate. The nanoDSP module can only tolerate defects up to 5% without incurring in a large performance loss at the output. For large d_e (e.g., $d_e = 3$), the nanoDSP module becomes overprotected with no noticeable performance

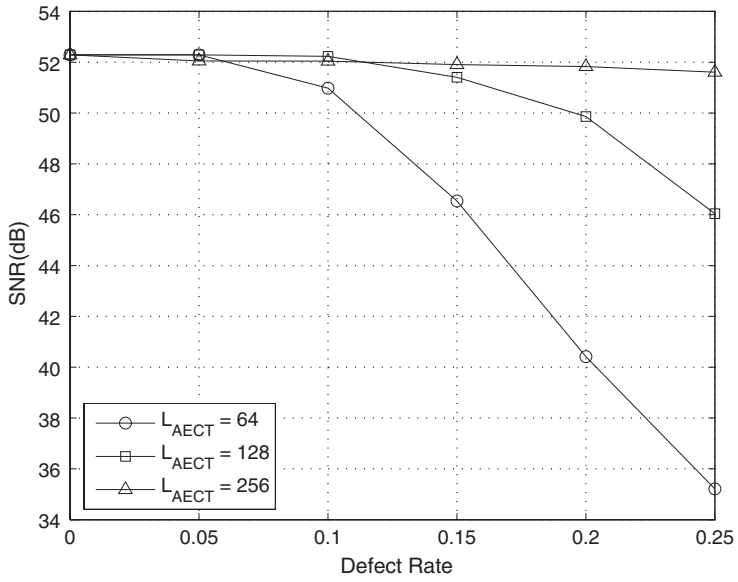

Fig. 9. Defect tolerance of different enhanced ECT sizes with $N = 16$, $SIR = 20dB$.

loss at very high defect rates. However, the ECT overhead is also significant because the size of the ECT increases exponentially with d_e. Therefore, it is important to select an appropriate d_e by carefully balancing defect and error tolerance with design overhead. When these factors are taken into account, $d_e = 2$ represents a reasonably good choice for designing this nanoDSP module for the selected application.

An enhanced ECT provides flexibility for fine-grained performance-overhead trade-offs. This is clearly shown in Figure 9, in which the enhanced ECTs with different sizes have smoother performance profiles than for the basic ECT. Therefore, we can refine the ECT size based on the system performance requirement for trade-offs. For example, if we want the output SNR to be about 51dB when the defect rate is 20%, then $L_{AECT} = 192$ is a good choice.

As discussed previously in Section 4, a higher level of error tolerance must be provided for the MSBs (rather than treating all bits equally). From an algorithmic viewpoint, this is an effective criterion to achieve reliable DSP-type computation. Consider the implementation of the filter with word length $N = 32$. We compare three designs with the same d_e. The first design is a direct implementation that provides $d_e = 4$ with no partitioning of long words. The other two designs partition the 32-bit word into two 16-bit subwords. One design provides equal protection to the two subwords, that is, by dividing the total $d_e = 4$ into $d_e = 2$ for each of two subwords. The other design provides $d_e = 3$ and $d_e = 1$ to the high-order and low-order subwords, respectively, so that error tolerance is algorithmically weighted towards the MSBs. Figure 10 shows the performance of these three designs. The direct implementation has the best error tolerance, that is, nearly no performance loss for defect rates up to 25%. However, the size of the ECT (measured as $L_{ep}(32, 4) = 41449$) is also the largest,

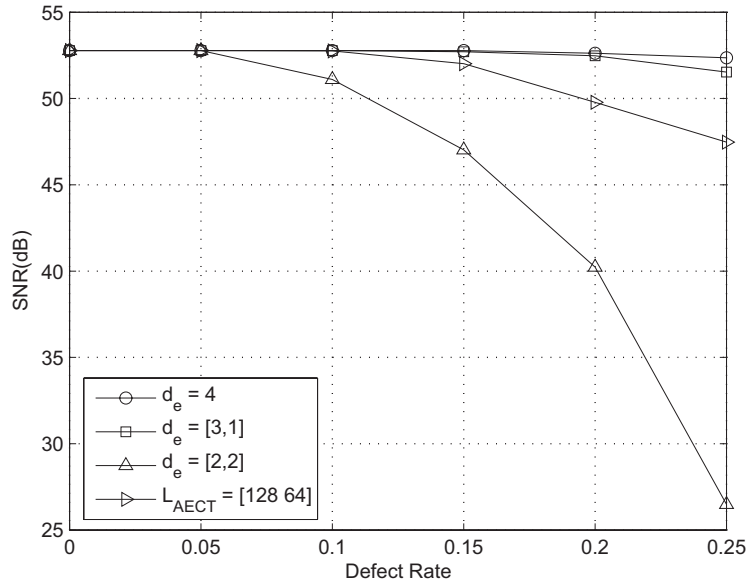

Fig. 10. Defect tolerance of different protection allocation schemes with $N = 32, SIR = 20dB$.

making this design impractical. The algorithmically weighted implementation achieves almost the same performance as the direct implementation with only a slight SNR degradation at a high defect rate of 25%. However, the overhead of the ECT is reduced significantly to only 357 words. The third implementation (using an equal protection scheme) has the smallest overhead ($L_{ep}(16, 2) = 137$ if the ECT is shared by the two subwords) but the worst performance. These results complement the design optimization procedure proposed in Section 4.

The enhanced ECT can further improve performance-overhead trade-offs. As shown in Figure 10, the enhanced ECT with the protection scheme $L_{AECT} = [128, 64]$ (i.e., employing two enhanced ECTs with size of 128 and 64 for the high-order and low-order subwords, respectively) can achieve comparable error tolerance performance as the basic ECT with $d_e = [3, 1]$ scheme, while utilizing less than 25% of the resources.

Usually the defect rate encountered once a nanosystem is assembled is different from the one assumed in its design. The difference in rates is referred to as a mismatch. To investigate the effects of a possible mismatch, a ECT for a defect rate $p = 10\%$ was utilized as reference by varying the defect rate in a range from zero to 25%. As shown in Figure 11, the enhanced ECT can tolerate a large range of mismatches with no noticeable performance degradation. Since we use the net contribution of an error pattern to decide whether this error pattern is included in the ECT, the mismatch does not incur in a significant change among the error patterns, that is, most of the error patterns in the ECT may still be qualified to correct the errors generated by different defect rates. Furthermore, the defective address multiplexer will diminish the impact of small difference in the ECT table as optimal error pattern assignment cannot be

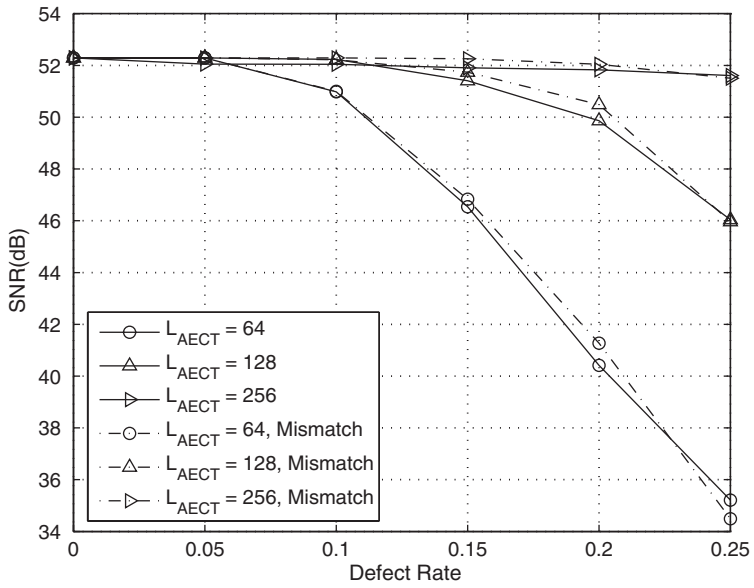

Fig. 11. Performance of the enhanced ECT subject to defect rate mismatches with $N = 16, SIR = 20dB$; and the ECT is designed with $p = 10\%$.

guaranteed at high defect rates. This feature indicates that the proposed technique is efficient and viable for a variety of scenarios in DSP applications.

Finally, we compare the proposed nanoDSP with conventional CMOS implementations. We assume the half pitch length (F) of nanowires is $3nm$ and the area of each crosspoint is $2F \times 2F$ [Stan et al. 2003]. The total area of main NW crossbar memory (16-bit word length) and address multiplexer is estimated to be $20\mu m^2$. Assuming $22nm$ process as the possible technology limit for CMOS, the ECT table (in CMOS with $d_e = 2$) consumes about $200\mu m^2$ using 6-transistor SRAM cells [Rabaey et al. 2004]. The area of address decoder implemented with NAND decoder using 2-input predecoders [Rabaey et al. 2004] is about $370\mu m^2$, and the area of a 32-bit carry lookahead adder is about $45\mu m^2$ by linearly scaled from the conventional carry lookahead adder [Lee et al. 1993]. Thus, the total area for the nanoDSP implementation of an FIR filter is about $650\mu m^2$. For the conventional CMOS implementation, the area of multipliers is dominant. Exploiting the symmetric property of the filter coefficients in a linear phase filter, the number of multipliers can be reduced to half of the number of the total coefficients, which becomes six in our study. Consider a conventional Booth multiplier with carry lookahead adders. The 16×16 multiplier takes about $400\mu m^2$ [Kang and Gaudiot 2006] assuming linear scaling to $22nm$. Therefore, the total estimated area of the FIR filter (including the latches and adder trees) implemented in conventional CMOS in $22nm$ process is about $2500\mu m^2$. Based on these results, the proposed nanoDSP system consumes only 25% of the area of the conventional CMOS in $22nm$. Other performance metrics are not available because the related study is still premature due to the current developments in nanowire technology.

6. CONCLUSION

In this article, we have proposed and analyzed a scheme that uses nanowire crossbars to design DSP nanosystems. A defect/error-tolerant nanoDSP architecture based on distributed arithmetic has been proposed to accomplish algorithmic error compensation. A comprehensive statistical study and simulation results have shown that the selection of an error correction capability (by word length and algorithmic significance) plays a significant role in achieving defect and error tolerance at a reduced overhead. Two ECT-based approaches have been proposed; the basic ECT approach enables low complexity and reasonable performance, while the enhanced ECT approach better accommodates error correction requirements of DSP applications under a wide range of operational scenarios. For a digital filtering DSP nanosystem, we have shown that the proposed approach can tolerate defect rates up to 25% without incurring in large system performance degradation and design overhead.

DSP integrated systems are a key component in a wide class of applications. Emerging nanotechnologies hold the promise for orders of magnitude improvement in performance and energy efficiency for signal processing. The proposed DSP nanosystem addresses the fundamental reliability challenge for computing at nano/molecular scales, thereby enabling efficient design for cost/performance and reliable operation to extend Moore's law beyond the limits of CMOS as predicted by the Technology Roadmap.

REFERENCES

DeHon, A. and Wilson, M. J. 2004. Nanowire-Based sublithographic programmable logic arrays. In *Proceedings of IEEE International Symponium on Field-Programmable Gate Arrays*. 123–132.

DeHon, A. 2005. Nanowire-Based programmable architechtures. *J. Emerging Technol. Comput. Syst. 1*, 109–162.

Goldstein, S. C. and Budiu, M. 2001. NanoFabrics: Spatial computing using molecular electronics. In *Proceedings of the 28th Annual International Symposium Computer Architecture*. 178–189.

Han, J. and Jonker, P. 2002. A system architecture solution for unreliable nanoelectronic devices. *IEEE Trans. Nanotechnol. 1*, 2001–2008.

Han, J. and Jonker, P. 2003. A defect- and fault-tolerant architecture for nanocomputers. *Nanotechnol. 14*. Institute of Physics Publishing, 224–230.

Heahth, J. R., Kuekes, P. J., Snider, G. S., and Williams, R. S. 1998. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Sci. 280*, 1716–1721.

Jeffery, C. M. and Figueiredo, R. 2006. Hierarchical fault tolerance for nanoscale memories. *IEEE Trans. Nanotechnol. 5*, 407–414.

Kang, J. and Gaudiot, J. 2006. A simple high-speed multipler design. *IEEE Trans. Comput. 55*, 1253–1258.

Koren, I., Koren, Z., and Stapper, C. H. 1994. A statistical study of defect maps of large area VLSI IC's. *IEEE Trans. VLSI Syst. 2*, 249–256.

Kuekes, P. J., Robinett, W., Seroussi, G., and Williams, R. S. 2005. Defect-Tolerant interconnect to nanoelectronic circuits: Internally redundant demultiplexers based on error-correcting codes. *Nanotechnol. 16*. Institute of Physics Publishing, 869–882.

Lee, Y. T., Park, I. C., and Kyung, C. M. 1993. Design of compact static CMOS carry look-ahead adder using recursive output property. *Electron. Lett. 29*, 794–795.

Lee, M., Kim, H. Y. K., and Choi, Y. H. 2004. A defect-tolerant memory architecture for molecular electronics. *IEEE Trans. Nanotechnol. 3*, 152–157.

Lee, K.-J., Hsieh, T.-Y., and Breuer, M. A. 2005. A novel test methodology based on error-rate to support error-tolerance. In *Proceedings of the IEEE International Test Conference*. 1136–1144.

MA, C., STRUKOV, D. B., LEE, J. H., AND LIKHAREV, K. K. 2005. Afterline for silicon: CMOL circuit architectures. In *Proceedings of IEEE Conference on Nanotechnology*. 175–178.

MACIAS, N. J. 1999. The PIG paradigm: The design and use of a massively parallel fine grained self-reconfigurable infinitely scalable architecture. In *Proceedings of the 1st NASA/DoD Workshop on Evolvable Hardwares*. IEEE Computer Society, 175–180.

MACIAS, N. J. AND ANTHANAS, P. M. 2007. Application of sefl-configurability for autonomous, highly-localized self-regulation. In *Proceedings of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems*. 397–404.

MANGER, D., SIPPER, M., STAUFFER, A., AND TEMPESTI, G. 2000. Toward self-repairing and self-replicating hardware: The embryonics approach. In *Proceedings of the 1st NASA/DoD Workshop on Evolvable Hardwares*. IEEE Computer Society, 205–214.

MIZAN, E. 2007. On protecting functional units with temporal redundancy. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 76–77.

NAEIMI, H. AND DEHON, A. 2008. Fault-Tolerant sub-lithographic design with rollback recovery. *Nanotechnol. 19*. Institute of Physics Publishing, 1–17.

NAIR, V. S. S. AND ABRAHAM, J. A. 1990. Real-Number codes for fault-tolerant matrix operations on processor arrays. *IEEE Trans. Comput. 39*, 426–435.

PARHI, K. K. 1999. *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley&Sons.

PELED, A. AND LIU, B. 1974. A new hardware realization of digital filters. *IEEE Trans. Acoust. Speech, Signal Process. 22*, 456–462.

RABAEY, J. M., CHANDRAKASAN, A., AND NICOLIC, B. 2004. *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice Hall.

RACHLIN, E. AND SAVAGE, J. E. 2008. A framework for coded computation. In *Proceedings of the IEEE International Symponium on Information Theory*. 2342–2346.

SPIELMAN, D. A. 1996. Highly fault-tolerant paralle computation. In *Proceedings of 37th Annual IEEE Conference on Foundation of Computer Science*. 154–163.

STAN, M. R., FRANZON, P. D., GOLDSTEIN, S. C., LACH, J. C., AND ZIEGLER, M. M. 2003. Molecular electronics: From devices and interconnect to circuits and architecture. *Proc. IEEE*, 1940–1957.

STRUKOV, D. B. AND LIKHAREV, K. K. 2007. Defect-Tolerant architectures for nanoelectronics crossbar memories. *J. Nanosci. Nanotechnol. 7*, 151–167.

SUN, F., FENG, L., AND ZHANG, T. 2008. Run-Time data-dependent defect tolerance for hybrid CMOS/nanodevice digital memories. *IEEE Trans. Nanotechnol. 7*, 217–222.

TAHOORI, M. B. 2005. A mapping algorithm for defect-tolerance of reconfigurable nano-architectures. In *Proceedings of the IEEE International Conference on Computer-Aided Design*. 668–672.

TANG, W. AND WANG, L. 2008. A DSP nanosystem with defect tolerance. In *Proceedings of the IEEE International Symposium on Nanoscale Architectures*. 32–37.

VON NEUMANN, J. 1956. Probabilistic logics and the synthesis of reliable organizms from unreliable compnents. In *Automata Studies*, Princeton University Press, 43–98.

WANG, T., QI, Z., AND MORITZ, C. A. 2004. Opportunities and challenges in application-tuned circutis and architectures based on nanodevices. In *Proceedings of the ACM Conference on Computeirn Frontier*. 503–511.

Received January 2009; revised April 2009; accepted June 2009 by Iris Bahar