Error-Correction and Crosstalk Avoidance in DSM Busses

Ketan N. Patel and Igor L. Markov

***Abstract*— Aggressive process scaling and increasing clock rates have made crosstalk noise an important issue in VLSI design. Switching on long, adjacent bus wires can lead to timing and logic faults. At the same time system-level interconnects have also become more susceptible to other less predictable forms of interference such as noise induced by power grid fluctuations, electromagnetic interference, and alpha-particle radiation. Previous work has treated these systematic and non-systematic forms of noise separately.**

We propose to make system level interconnects more robust using encoding that simultaneously addresses error-correction requirements and crosstalk noise avoidance. This is more efficient than satisfying these requirements separately. We give algorithms for obtaining optimal encodings, and present a practical class of codes called boundary-shift codes. We evaluate the overhead of our method, and make comparisons to using error-correction with simple shielding.

***Index Terms*— bus-encoding, error-correction, crosstalk, DSM busses**

I. Introduction

As device densities and clock rates continue to increase in VLSI circuits, bus crosstalk is becoming an increasingly important factor in performance optimization, and correlates with particular switching patterns on the bus. For example, simultaneous rising transitions on adjacent wires are sped up by noise, possibly leading to a hold violation. A rising transition on one wire can cause a neighboring wire to falsely transition, and lead to a logic fault. However, the most detrimental switching pattern on two neighboring wires is opposite transitions because both transitions slow down, potentially leading to a setup violation. Crosstalk can be mitigated by specialized routing strategies [4], intentionally skewing signal transition timings on adjacent wires [6], both active and passive shielding [7], [9], and signal encoding to minimize conflicts [13], [2].

Busses can also suffer from interference due to power-grid fluctuations, electromagnetic noise, or alpha-particle radiation. Many such effects, e.g., those caused by the simultaneous switching of many gates, are difficult to predict or prevent. They aggravate as the number of gates supplied by the same power grid increases, and may not correlate with particular switching patterns on the bus. Another source of faults is manufacturing defects. In the nanotechnology context, where circuits are manufactured with a significant proportion of faults, occasional errors may be unavoidable. Hence, preventive techniques may be insufficient, and active error-correction may be required.

Researchers previously studied error-correction for busses to combat crosstalk [3] and non-systematic interference [11], [1]. However, those methods only correct errors after errors occur rather than attempting to prevent the interference. Conversely, other bus-encoding techniques have been used to prevent crosstalk, but do not correct errors [13], [2]. For example, Victor and Keutzer [13] have proposed encoding the bus to prohibit opposite transitions on adjacent wires. They call such codes self-shielding as they are an alternative to shielding wires.

We combine the goals of providing self-shielding and error-correction in designing the bus-encoding. This is more effective than the conventional approach of treating these separately. Since our method not only reduces the interference due to crosstalk, but also corrects errors, it can be useful in a variety of applications including nanotechnology, low-swing signaling, and radiation-hardened circuits. We provide algorithms for generating optimal bus-encodings, and present a general construction method for a practical class of codes. Encoding and decoding circuits are given for specific codes.

An earlier version of this work was presented at the International Workshop on System-Level Interconnect Prediction, SLIP 2003.

K. N. Patel is with Qualcomm Corp. and I. L. Markov is with the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: knpatel@eecs.umich.edu; imarkov@eecs.umich.edu).

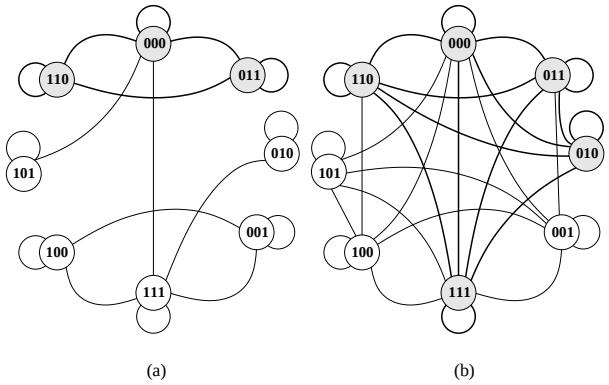

Fig. 1. Graphical representation of optimal 3-bit self-shielding codes: (a) single-error-detecting, (b) no error control.

II. Notation and Terminology

Our goal is to design a bus-encoding scheme that avoids crosstalk while simultaneously providing error-correction. Victor et al. [13] consider a related problem, and we follow much of their terminology. We also use some basic background in error-correction [10], [12] and self-checking [14], [8] techniques.

We model the bus as an n-bit communication channel. During each *signaling interval* the encoding scheme is used to select and transmit an n-bit word, called a *codeword*, from a possibly dynamic set called a *codebook*. The codebook is dynamic in the sense that it can be a function of previously transmitted codewords. We call the overall encoding scheme a *code*. A code is *memoryless* if it uses a fixed codebook. A code's *rate* is $\log_2 |C_{min}|$, where $|C_{min}|$ is the number of codewords in the smallest codebook. This is the minimum number of bits that can be encoded during every signaling interval.

We say a pair of codewords contains an *invalid transition* if transitioning from one codeword to the other causes adjacent bits to switch in opposite directions. For example, the following codewords contain an invalid transition by bits 3 and 4:

$$\begin{array}{ll} c_1 \rightarrow & 0\,1\,\mathbf{1}\,\mathbf{0}\,0\,0\,1\,1 \\ c_2 \rightarrow & 1\,1\,\mathbf{0}\,\mathbf{1}\,1\,0\,1\,0 \end{array}$$

Such transitions are undesirable because they increase crosstalk noise. A code is *self-shielding* if it does not allow invalid transitions; this terminology comes from the conventional technique of using shielding wires to prevent crosstalk. Following Victor et al. [13], we assume that neighboring wires are routed in parallel. Therefore, the encoded bus is effectively self-shielding.

In addition to avoiding invalid transitions, we want to be able to differentiate codewords reliably even in the presence of errors (bit flips). This is possible if the codewords in the codebook have a large enough *Hamming distance* between them, that is, if they differ in enough bit positions. For example, if the codewords have a minimum Hamming distance of three between them, any single error is correctable since the "noisy" codeword must be closer to the original codeword than to any other. In general, if the minimum Hamming distance between any two codewords is d, then we can either correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors, or detect up to $d-1$ errors [10].

A binary code is *linear* if the bitwise sum (mod 2) of any two codewords is also a codeword. A linear code can be represented by an

	Without Memory		With Memory		
Wires	self-shielding	single-error-correcting self-shielding	self-shielding	single-error-correcting self-shielding	single-error-correcting boundary-shift
3	2.32	1.00	2.32	1.00	1.00
4	3.00	1.00	3.17	1.00	-
5	3.70	1.59	3.91	2.00	2.00
6	4.39	2.32	4.75	2.32	-
7	5.09	2.58	5.52	3.17	3.00
8	5.78	3.17	6.34	3.59	-
9	6.48	3.81	7.14	4.25	4.00

TABLE I

MAXIMUM RATE FOR SELF-SHIELDED CODES. THE FIRST COLUMN GIVES THE NUMBER OF WIRES USED FOR THE ENCODING. THE REMAINING COLUMNS GIVE THE MAXIMUM NUMBER OF BITS THAT CAN BE ENCODED BY THE SELF-SHIELDING CODES SPECIFIED BY THE COLUMN HEADINGS.

independent set of codewords [12]. All other codewords can then be formed by a linear combination of these. A standard representation of a linear code is the *generator matrix*, whose rows are an independent set of codewords. For example, a generator matrix for the code $\{0000,\ 0011,\ 0101,\ 0110,\ 1001,\ 1010,\ 1100,\ 1111\}$ is

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

The generator matrix maps information bits to codewords: multiply the generator matrix by a row-vector of information bits. For example, G maps [1 1 1] to the codeword [1 0 0 1]. An n-bit linear code that encodes k bits and has minimum Hamming distance d between all codewords is called an $[n, k, d]$ code.

III. MEMORYLESS CODES

In a memoryless self-shielding code, there is a fixed codebook, and any codeword may be transmitted after another. Therefore, all transitions between codewords must be valid. This can be captured by a graphical model [13], with each n-bit word represented by a vertex in the graph, and each valid transition between words by an edge between the corresponding vertices (see Figure 1b). A memoryless self-shielding code then consists of a set of vertices forming a *clique*, that is, a set of vertices with edges between every pair in the set. The maximum-rate code is the largest clique in the graph. One such clique is highlighted for the graph in Figure 1b. A closed-form for the size of the largest clique for general n can be derived [13].

This graphical model can accommodate additional constraints. In our case the Hamming distance between any two codewords must be at least d. This can be incorporated into the model by only placing edges between vertices if the corresponding words satisfy both the Hamming distance and valid transition constraints. Figure 1a shows such a graph for a single-error-detecting ($d = 2$), self-shielding code on three wires with a maximum-size clique highlighted.

While the max-clique problem is NP-complete [5, pp. 53-56], it can be solved for small graphs in practice. We used a simple pruning algorithm (see Algorithm 1), coded in MATLAB, that determines if a k-clique exists in the graph. For the $n = 9$ case, the algorithm required approximately 45 minutes on a Sun-Sparc. The results are shown in Table I. Results for memoryless codes without error-correction (column two) match those given by Victor et al. [13].

IV. CODES WITH MEMORY

In a code with memory, the codebook may be a function of the previously transmitted codeword. We represent such codes with two graphs. The first graph G_1 has a vertex for each n-bit word, and an edge between two vertices if they form a valid transition. The second graph G_2 contains the same vertices as G_1, but has edges between vertices if the Hamming distance between them is greater than $d-1$.

```
function k-clique(G, k)
{   V = {vert. of G sorted by ↑ degree}
    if (G is complete & |V|≥k)
        return 1
    for (v ∈ V)
    {   remove vert. w/ degree < k from G & V
        if |V| < k return 0
        G_s = {subgraph w/ vert. incident to v}
        if (k-clique(G_s, k)==0)
           remove v from G & V
        else
           return 1
    }
}
```

Algorithm 1: Solving Max-Clique

Intuitively, G_1 and G_2 represent the self-shielding and the Hamming distance constraints, respectively.

In a self-shielding minimum-distance-d code with rate $\log_2 M$, each codeword can transition to a size-M subset of codewords with a Hamming distance $\geq d$ between them. In our graphical representation, each vertex in graph G_1 must have edges to at least M vertices that form a clique in graph G_2. We can determine if such a code exists by iteratively eliminating vertices that do not meet this condition from both graphs. If a non-empty set of vertices remains, then it forms a code with the desired properties, otherwise none exists. The pseudo-code is shown as Algorithm 2.

```
function Exist_Code(G1, G2, k)
{   V = {vert. of G1}
    Vin = {}
    while (V not empty)
    {   if (V==Vin)
           return 1
        v = vertex from {V-Vin} of lowest degree
        G_s = {subgraph of G2 w/ vertices
              incident to v in G1}
        if (k-Clique(G_s, k)==0)
           remove v from G1, G2 & V
           Vin = {}
        else
           add v to Vin
    }
    return 0
}
```

Algorithm 2: Search for code with memory

This algorithm makes a series of calls to k-clique, and the runtime for specific cases varies greatly for different implementations of k-clique. We calculated the optimal codes for $n \leq 8$ using Algorithm 1, but for $n = 9$ we used an integer linear programming (ILP) formulation for the max-clique problem. A binary variable was assigned to each vertex in the graph with a one denoting membership in the clique. For every pair of vertices not sharing an edge, we imposed the constraint that both cannot be members of the clique. The size of the max-clique was found by maximizing the sum of the

variables, i.e., the number of members in the clique. This optimization was performed by the commercial CPLEX program in several hours on a Sun-Sparc workstation. While the ILP-based computation was more efficient than Algorithm 1 for $n = 9$, for cases in Section III it was 5 times slower. The overall results are shown in Table I. Those in columns two and four, for codes without error-correction match results from [13].

V. Boundary-Shift Codes

The algorithm presented in the previous section finds maximum-rate codes, but not encoders or decoders for them. Decoding may require particularly significant resources. Further, the algorithm itself becomes computationally infeasible for large busses. In this section, we construct a class of practical codes with necessary circuits.

We define a *dependent boundary* in a word as a place where two adjacent bits differ, and denote it by the position of the leftmost bit of the boundary. Two words sharing no dependent boundaries, cannot form an invalid transition. For example, consider the words:

$$\begin{array}{ll} c_1 \rightarrow & 0\,1\,1\,0\,0\,1\,1\,1 \\ c_2 \rightarrow & 1\,1\,0\,0\,1\,1\,1\,0 \end{array}$$

Here c_1 and c_2 have dependent boundaries $\{1, 3, 5\}$ and $\{2, 4, 7\}$, respectively. Since there is no overlap, the transition must be valid.

If a codebook has codewords with only even dependent boundaries, then performing a 1-bit circular right shift yields a new codebook with no even dependent boundaries. Since the two codebooks share no dependent boundaries, we can alternate between the two to obtain a self-shielding code. We call this a *boundary-shift code*.

This construction requires an error-correcting code with no odd dependent boundaries. Let C be an $[n, k, d]$ code, and let C' be formed by duplicating each bit position in C. Then C' is a $[2n, k, 2d]$ code with no odd dependent boundaries, since every bit in an odd bit position is followed by a copy. By alternating between C' and a shifted version of it, we obtain a $[2n, k, 2d]$ self-shielding code. In addition, *puncturing* C' in the last bit position, i.e., removing the last bit in every codeword, yields a $[2n - 1, k, 2d - 1]$ code.

A single-parity-check code used in the above construction gives an infinite class of single-error-correcting codes. First a $[k + 1, k, 2]$ single even parity-check code is formed by appending a parity bit to k data bits. Then applying our construction, yields a $[2k + 1, k, 3]$ single-error-correcting self-shielding code.

Consider a "$[9, 4, 3]$" boundary-shift code with generator matrices:

$$G_0 = \begin{bmatrix} \mathbf{11}\ 00\ 00\ 00\ \mathbf{1} \\ 00\ \mathbf{11}\ 00\ 00\ \mathbf{1} \\ 00\ 00\ \mathbf{11}\ 00\ \mathbf{1} \\ 00\ 00\ 00\ \mathbf{11}\ \mathbf{1} \end{bmatrix} \quad G_1 = \begin{bmatrix} \mathbf{1}\ \mathbf{11}\ 00\ 00\ 00 \\ \mathbf{1}\ 00\ \mathbf{11}\ 00\ 00 \\ \mathbf{1}\ 00\ 00\ \mathbf{11}\ 00 \\ \mathbf{1}\ 00\ 00\ 00\ \mathbf{11} \end{bmatrix}$$

Generator matrices G_1 and G_0 are used for encoding during odd and even clock cycles respectively. Equivalently, G_0 can be used for all cycles, with the output then right shifted for odd cycles. The following example illustrates how this code would be used to encode a 4-bit bus. The intermediate *pre-shifted* output is shown for clarity.

time	input		pre-shifted output		output
0	1 0 1 0	→	1 1 0 0 1 1 0 0 0	→	1 1 0 0 1 1 0 0 0
1	0 1 1 1	→	0 0 1 1 1 1 1 1 1	→	1 0 0 1 1 1 1 1 1
2	1 0 0 0	→	1 1 0 0 0 0 0 0 1	→	1 1 0 0 0 0 0 0 1
3	0 1 0 0	→	0 0 1 1 0 0 0 0 1	→	1 0 0 1 1 0 0 0 0

We duplicate each input bit, and append the parity check at the end, yielding the pre-shifted output. If the clock cycle is odd, we perform a 1-bit circular right shift before transmitting.

At the receive side, we first undo the right shift if the clock cycle is odd. Decoding is then done by majority vote, where the two "copies" of the desired bit are augmented by a third generated by the sum (mod 2) of one copy of each of the other information bits and the parity check. For example, in an even cycle three independent copies of the first information bit are given by y_0, y_1 and $(y_2 + y_4 + y_6 + y_8)$ mod 2. Since a single error will affect at most one of the three copies, it is correctable. The following example shows how noisy versions of the codewords in the previous example would be decoded after unshifting. The highlighted bits correspond to errors.

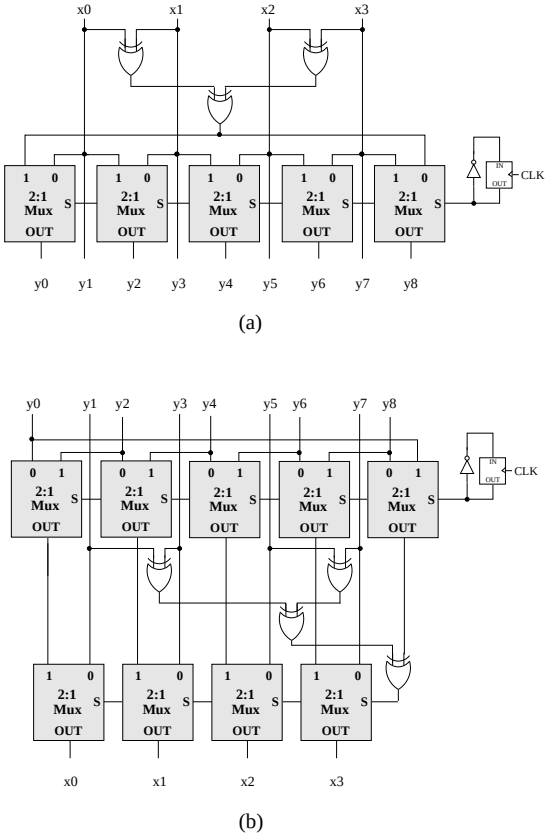

Fig. 2. Encoder (a) and decoder (b) for "$[9, 4, 3]$" boundary-shift code.

noisy output		majority vote		data
1 **0** 0 0 1 1 0 0 0	→	(1**0**1) (000) (111) (000)	→	1 0 1 0
0 0 1 1 1 1 1 1 **0**	→	(00**1**) (11**0**) (11**0**) (11**0**)	→	0 1 1 1
0 1 0 0 0 0 0 0 1	→	(**0**11) (00**1**) (00**1**) (00**1**)	→	1 0 0 0
0 **1** 1 1 0 0 0 0 **0**	→	(0**11**) (11**0**) (00**1**) (00**1**)	→	**1** 1 0 0

The last codeword is decoded incorrectly as it contains two errors, and is therefore beyond the code's error-correcting capability.

Table I compares these codes to optimal ones. Their rates are better than those of optimal memoryless codes, and comparable to those of optimal codes, particularly when integer rates are of interest.

VI. Practical Considerations

The codes constructed in the previous section are based on very simple error-correcting codes, and can be encoded and decoded efficiently. Figures 2a and 2b show an encoder and decoder respectively for the $[9, 4, 3]$ code. These circuits generalize in a straightforward manner for larger single-error-correcting codes. Gate counts and maximum circuit depths are given in Table III for a range of bus sizes along with closed form expressions for the general case.

For large bus sizes the increased circuit depth may lead to significant delay. This problem can be solved by breaking the bus into smaller sub-busses separated by shielding wires. While slightly increasing the number of wires and gates needed, this limits the circuit depth. It also increases the error-correction capability, since single errors in each sub-bus can then be corrected independently.

Code	Advantages	Drawbacks
Optimal memoryless	- encoding/decoding by combinational circuits	- relatively low rate - code construction difficult - decoder may be complex
Optimal with memory	- maximum rate	- code construction difficult - encoder/decoder may be very complex - possible error propagation
Boundary-Shift Codes	- achieve higher rate than opt. memoryless codes - scalable construction - simple encoder/decoder - integer rates; systematic	- achieve slightly lower rate than optimal codes

TABLE II

COMPARISON OF BOUNDARY-SHIFT CODES AND OPTIMAL SELF SHIELDING ERROR-CORRECTING CODES.

Method	Advantages	Drawbacks
Shielding Wires	- crosstalk prevention - simple construction - no additional delay	- no error-correction - additional wires
Self-Shielding Codes [2], [13]	- crosstalk prevention - relatively high rate	- no error-correction - encoding/decoding logic - additional wires & delay
Error-Correcting Codes [1], [3], [11]	- error-correction	- no crosstalk prevention - encoding/decoding logic - additional wires & delay
Bus Precharging	- crosstalk prevention - no additional wires	- no error-correction - high power consumption
Proposed Boundary-Shift Codes	- crosstalk prevention - error-correction	- encoding/decoding logic - additional wires & delay

TABLE IV

COMPARISON OF BOUNDARY-SHIFT CODES AND OTHER SHIELDING AND ERROR CORRECTION METHODS.

While the bus-encoding is designed to correct soft errors that occur on the bus, it has the beneficial side-effect of also correcting most single stuck-at faults in the encoding and decoding circuits. Specifically, with the exception of stuck-at faults on the flip-flops used for synchronization, any single stuck-at fault in the encoding or decoding circuit leads to at most a single error, which is then automatically corrected because of the encoding. A fault in the synchronization flip-flops can cause uncorrectable errors, and therefore should be more carefully protected against. Particular care should be taken to ensure the inverter between the flip-flop input and output has a large enough delay to avoid hold time violations.

Self-shielding error-correcting codes do not necessarily remain self-shielding in the presence of errors. A bus error may cause an invalid transition, while such transitions are prevented in error-free operation. However, even in this case, assuming that the error occurs equiprobably anywhere on the bus line, the effective coupling capacitance between the affected lines would be 50% lower on average than for an unprotected bus, significantly mitigating the resulting crosstalk.

A potentially useful feature of the proposed codes is that they are *systematic*, i.e., the information bits are embedded in the encoded codeword, and hence can be obtained without decoding logic. For example, the information bits of the $[9, 4, 3]$ code are given by bits y_1, y_3, y_5 and y_7. Of course, simply using these bits rather than decoding the full codeword sacrifices the code's error-correction capabilities. However, error-correction may only be necessary for certain distant bus pins, while remaining pins may use error-detection or simply the embedded information bits.

In the proposed method, we assume that neighboring wires on the bus are routed in parallel. Therefore, automatic routing of busses with the proposed encoding must be used with care because of the wire ordering requirement. If the wire ordering is not preserved in the routing, the self-shielding properties of the encoding may be lost, though the error-correction properties will be unaffected.

A possible concern in codes with memory is that since future codebooks may depend on the current transmitted codeword, an uncorrectable error may propagate and cause additional errors. A clear advantage of boundary-shift codes is that they are not vulnerable to error propagation, since the codebook only depends on the time index. As long as the source and destination are synchronized, no error propagation will occur.

Bus Size	Wires	Encoder		Decoder	
		Gates	Circuit Depth	Gates	Circuit Depth
4	9	10	3 gates	15	4 gates
8	17	18	4 gates	27	5 gates
16	33	34	5 gates	51	6 gates
32	65	66	6 gates	99	7 gates
64	129	130	7 gates	195	8 gates
n	$2n+1$	$2n+2$	$\lceil \log_2 n \rceil + 1$	$3n+3$	$\lceil \log_2 (n+1) \rceil + 1$

TABLE III

ENCODER/DECODER GATE COUNTS AND CIRCUIT DEPTHS FOR SINGLE-ERROR-CORRECTING BOUNDARY-SHIFT CODES.

Table II summarizes advantages and drawbacks of several codes. Boundary-shift codes have a higher rate (i.e., require fewer additional wires) than optimal memoryless codes, and only a slightly lower rate than optimal codes. Their primary advantages are a simple, scalable construction as well as practical encoders and decoders.

In Table IV we contrast boundary-shift codes with other shielding and error-correction methods. Compared to self-shielding codes from [13], our codes are not only able to prevent crosstalk but also tolerate errors; this additional protection comes at the cost of rate reduction (see Table I). Compared to shielding, our technique provides error-correction in addition to self-shielding without appreciably reducing the rate, though some encoding and decoding logic is required. Our approach is more effective than treating error-correction and shielding separately. *For example, to protect a 4-bit bus from single errors requires a total of at least 7 bits [10]. Adding shielding wires to this encoded bus to prevent crosstalk then results in a 13-bit bus. In comparison, the "*$[9, 4, 3]$*" single-error-correcting boundary-shift code illustrated in Section V achieves the same error protection for the 4-bit bus using only 9 bits.* In general, for a bus of size n the separate optimal coding and shielding approach requires $2 \log_2 n$ more wires than the proposed approach. Furthermore, since the proposed boundary-shift codes are based on simpler codes, they have lower encoding and decoding complexities than the separate approach; the complexity gap between the two approaches grows with increasing bus size. Precharging busses to prevent crosstalk is also an alternative, however it can be costly in terms of power consumption, and does not provide error-correction.

VII. CONCLUSIONS

We have considered the problem of designing bus-encoding schemes that provide both crosstalk prevention and active error-correction. The former reduces crosstalk interference, while the latter corrects faults after they occur, regardless of their origin. This is an important improvement over previous methods, which have been designed to either reduce crosstalk interference or provide error-correction but not both. Our joint approach is particularly applicable to scenarios, such as nanotechnology and radiation hardened circuits for satellite communications, where random errors are a concern in addition to crosstalk interference. We give algorithms for finding optimal codes for various constraints and code parameters. An important contribution of this paper is a practical class of error-correcting self-shielding codes called boundary-shift codes. These codes are derived from conventional error-correcting codes, which have been studied extensively in the literature. For the specific case of single-error-correcting boundary-shift codes we give gate level encoding and decoding circuits.

REFERENCES

[1] D. Bertozzi, L. Benini, and B. Ricco. Energy-efficient and reliable low-swing signaling for on-chip buses based on redundant coding. *Proc. ISCAS*, pp. 93–96, 2002.

[2] C. Duan, A. Tirumala, and S. P. Khatri. Analysis and avoidance of cross-talk in on-chip buses. *Hot Interconnects 9*, pp. 133–138, Aug. 2001.

[3] M. Favalli and C. Metra. Bus crosstalk fault-detection capabilities of error-detecting codes for on-line testing. *IEEE Trans. on VLSI*, pp. 392–396, Sept. 1999.

[4] T. Gao and C. L. Liu. Minimum crosstalk channel routing. *Proc. ICCAD*, pp. 692–696, Nov. 1999.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[6] K. Hirose and H. Yasuura. A bus delay reduction technique considering crosstalk. *Proc. DATE*, pp. 441–445, 2000.

[7] H. Kaul, D. Sylvester, and D. Blaauw. Active shields: A new approach to shielding global wires. *Proc. GLS-VLSI*, pp. 112–117, Apr. 2002.

[8] P. K. Lala. *Self-Checking and Fault-Tolerant Digital Design*. Academic Press, Inc., 2001.

[9] K. M. Lepak, I. Luwandi, and L. He. Simultaneous shield insertion and net ordering under explicit RLC noise constraint. *Proc. DAC*, pp. 199–202, June 2001.

[10] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1996.

[11] D. Rossi, V. van Dijk, R. Kleihorst, A. Nieuwland, and C. Metra. Coding scheme for low energy consumption fault-tolerant bus. *Proc. IEEE On-line Testing Workshop*, pp. 8–12, 2002.

[12] S. A. Vanstone and P. C. van Oorschot. *An Introduction to Error Correcting Codes with Applications*. Kluwer, 1989.

[13] B. Victor and K. Keutzer. Bus encoding to prevent crosstalk delay. *Proc. ICCAD*, pp. 57–69, Nov. 2001.

[14] J. F. Wakerly. *Error detecting codes, self-checking circuits and applications*. Elsevier North-Holland, Inc., 1978.