

# A Unified Transformational Approach for Reductions in Fault Vulnerability, Power, and Crosstalk Noise & Delay on Processor Buses

Raid Ayoub

University of California at San Diego  
CSE Department  
rayoub@cs.ucsd.edu

Alex Orailoglu

University of California at San Diego  
CSE Department  
alex@cs.ucsd.edu

## Abstract

In this paper we propose a coding scheme for general-purpose applications that can reduce power dissipation, crosstalk noise and crosstalk delay on the bus lines while simultaneously detecting errors at run time. The reduction in power dissipation can be achieved through reducing the bus switching activity. Not only is the switching activity in individual lines reduced but so is the coupling activity across the adjacent lines, the major contributor to the overall power dissipation in deep submicron technology. Detailed analysis of crosstalk noise and delay shows that eliminating certain patterns of transitions and reducing the infeasible ones in terms of crosstalk noise and power dissipation is a feasible strategy for alleviating these problems. We propose an encoding technique consisting of the use of predefined patterns of transitions, one for each possible combination of input data, to generate the codewords. The restriction to the predefined patterns of transitions enables fast encoding and low hardware overhead. This work presents an extensive analysis of the consequent reduction in crosstalk and power. SPICE derived experimental results show a reduction in worst case crosstalk delay and noise, ranging up to 24% and 10% respectively. Extensive experimental results for various applications show significant reduction in power dissipation ranging up to 44% for switching activity on the bus lines and up to 25% for coupling activity. The results also show a drastic reduction ranging up to 98% in the number of patterns that are most likely to produce crosstalk errors.

## 1. Introduction

The unprecedented demand for faster and more powerful processors and embedded systems increasingly necessitates the use of deep-submicron technology. Scaling down into deep-submicron has led to enlargement in processor and embedded system integration. However, increases in the level of integration lessen the horizontal spacing between interconnects, resulting in the dominance of the coupling capacitors between adjacent lines. The supremacy of coupling capacitors has in turn elevated crosstalk noise and delay into significant design challenges and has also increased power dissipation on the bus. Crosstalk noise and delay in turn dwindle the reliability of data transfers across the bus lines and also limit overall performance. Power dissipation has become an important issue in recent years, driven by the widespread demand for hand-held and wireless devices. Diminished power dissipation leads to an extended battery life, affordable increases in die sizes and reductions in the possibility of thermal defects on the chip.

A variety of techniques have been proposed to ameliorate the aforementioned problems associated with deep-submicron technology. Some of these techniques attempt to manage these problems at the design level while others do so at run time. An illustrative technique, the *Bus Invert* scheme, was proposed to reduce power dissipation in the bus lines due to self transitions [1].

Subsequently, the *Calculated Odd/Even Bus Invert* scheme has been proposed to reduce power dissipation due to coupling transitions [2]; the technique bears significant similarity to the *Bus Invert* scheme. The *Calculated Odd/Even Bus Invert* scheme has reduced coupling transitions by 36%, but the ensuing hardware complexity limits the applicability of this scheme. Another encoding scheme [3] has been proposed to reduce power dissipation by reducing inter-wire transitions. However, the inherent complexity of the associated encoder and decoder structures sharply limits its applicability. Among the techniques that are proposed to handle crosstalk at the design stage is the *shielding method* [4]. This method is implemented by adding a VDD and VSS “shield” between the data lines. The overhead of this technique is considerable as it doubles area, certainly not an insignificant cost. Other techniques to reduce crosstalk, like transistor sizing [5], and buffer insertions [6,7] have been proposed. Techniques for handling crosstalk at runtime have also been suggested using schemes like on-line detection with recovery [8,9]. In the domain of application specific embedded processors, it has been shown that utilizing application specific knowledge can constitute the keystone of a set of techniques aimed at improved power savings, reliability and performance [10,11].

In this paper, we present a coding scheme for general purpose processor buses to tackle the aforementioned problems that are associated with deep-submicron technology. Extensive analysis that we have conducted indicates that eliminating certain patterns of transitions on the bus lines can help abate the worst case crosstalk noise and delay and provide the capability of capturing on-line crosstalk errors. Abating the power dissipation can be achieved through reducing the transition activity on the bus. These problems can be ameliorated through encoding the data words in the form of transitions instead of values. In this work we propose a coding scheme that maps the data words to prespecified patterns of transitions that deliver power savings, abating worst case crosstalk noise and delay and providing the capability of detecting crosstalk faults at run time. Also we provide an extensive analysis of the proposed coding scheme. Experimental results confirm the theoretically expected benefits of using the coding scheme.

This paper is organized as follows. In section 2, we review the coupling model for crosstalk noise and delay. In section 3, we report on experimental data obtained through SPICE simulations for crosstalk noise and delay. In section 4, we review the power model for self and coupling transitions. In section 5, we present the objectives of our research. In section 6, we introduce the proposed coding scheme. In section 7, we discuss the power savings possible by using the proposed coding scheme. In section 8, we show how the proposed coding scheme can be used for on-line error detection with recovery and also report on a set of results regarding the reduction in the probability of errors using our coding scheme. A brief set of conclusions recaps the paper in section 9.

\* This work is supported in part by NSF Grant 0082325.

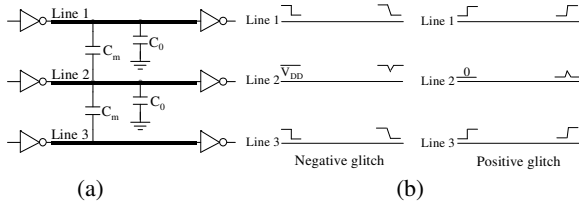


Figure 1: Two different cases of crosstalk noise

## 2. Coupling model for crosstalk noise and delay

We illustrate the coupling effect of crosstalk by considering a simple case of three parallel lines with inverters as drivers, as shown in Figure 1(a). In general, the capacitor is inversely proportional to the line spacing and proportional to the length of the line that runs in parallel. The reduction in the horizontal spacing between the lines results in the domination of the cross coupling capacitor,  $C_m$ .

Crosstalk noise occurs when an active line (aggressor) induces an undesired noise in a quiet line (victim). In the case of three lines, the crosstalk noise worst case scenario could occur either when the signal on the victim line is steady high and the signals on the aggressors switch from high to low ( $\downarrow H \downarrow$ )<sup>1</sup> or when the signal on the victim line is steady low and the signals on the neighboring lines are switched from low to high ( $\uparrow L \uparrow$ ) as shown in Figure 1(b). In the case of crosstalk error occurrence in the victim line, these vectors will be received as ( $\downarrow \downarrow \downarrow$ ), and ( $\uparrow \uparrow \uparrow$ ), instead respectively.

Now we discuss the effect of crosstalk on delay; we denote this as *crosstalk delay*. The bus line delay is proportional to the product of the resistance and capacitance of the bus line. The capacitance of the bus line consists of self capacitance,  $C_0$ , and coupling capacitance,  $C_m$ , as shown in Figure 1(a). Eq. 1 approximates the effective capacitance equation for the center line in Figure 1 [11],

$$C_{eff} = C_0 + C_m \left| \frac{\Delta V_2 - \Delta V_1}{E} \right| + C_m \left| \frac{\Delta V_2 - \Delta V_3}{E} \right| \quad (1)$$

where  $C_{eff}$  denotes the effective capacitance of the center line,  $\Delta V_i$  the voltage deviation in line  $i$ , and  $E$  the power supply voltage. This equation shows that  $C_{eff}$  depends on the switching behavior of the center line and its neighbors. Using this equation, it can be seen that  $C_{eff}$  attains the maximum value of  $C_{eff} = C_0 + 4C_m$  when the three lines switch simultaneously with the center line switching in a direction opposite to its neighbors ( $\uparrow \downarrow \downarrow$ ) or ( $\downarrow \uparrow \uparrow$ ), which maximizes the crosstalk delay as well.

## 3. Crosstalk noise and delay on the bus lines

To determine the actual values of crosstalk noise and delay on the bus lines, we have used an extensive set of SPICE simulations. Appendix A shows the configurations used for SPICE simulations. Figure 2 shows the variation in the crosstalk amplitude when one to five aggressors are switching on a bus of six lines. The results show that the crosstalk noise associated with one aggressor is significantly lower in comparison to the cases when multiple aggressors exist, indicating that the crosstalk induced errors are most likely to occur in the case of multiple aggressors. Figure 3 shows the effect of the transition activities on the crosstalk

<sup>1</sup> We use the symbol ( $\uparrow$ ) to represent the low-to-high transition on the bus line, ( $\downarrow$ ) to represent the high-to-low transition on the bus line, ( $\updownarrow$ ) to represent either transition event on the bus line, (H) to represent a line that is steady high, and (L) to represent a line that is steady low.

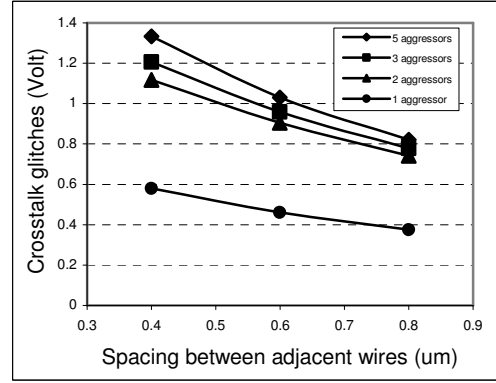


Figure 2. Crosstalk noise versus spacing between adjacent lines

delay for a bus of three lines. Case A represents the transitions  $\downarrow \uparrow \downarrow$ , case B represents the transitions  $\downarrow \uparrow L$ , and case C represents the transitions  $L \uparrow L$ . The results show that the worst case crosstalk delay, case A, is significantly higher than the other cases. Using the results in both Figures 3 and 4, one can see that eliminating the simultaneous transitions in three adjacent lines reduces both worst case crosstalk noise and delay on the bus lines.

## 4. Power model for self and coupling transitions

The average power dissipation in the bus lines is a result of the power dissipation in self-transitions and the power dissipation in the coupling transitions. *Self-transitions* can be defined as transitions on the self-capacitance  $C_0$ , whereas *coupling transitions* can be defined as transitions on the coupling capacitor  $C_m$ . The average power dissipation on the bus is given by [2]:

$$P_{avg} = \frac{1}{2} (\alpha_s C_0 + \alpha_c C_m) \cdot V_{dd}^2 \cdot f \quad (2)$$

where  $\alpha_s$  and  $\alpha_c$  represent the number of average self transitions ( $n_s$ ) and coupling transitions ( $n_c$ ) in one cycle, respectively. For the case of self-transitions,  $n_s = 1$  for both transitions  $\uparrow$  and  $\downarrow$ . For the case of coupling transitions, three different values are possible for  $n_c$ ; 0 (in the case of no transitions activity as shown in Figure 4 (a)), 1 (in the case of a charging or discharging event, as shown in Figure 4(b&c)) and 4 (in the case of toggling event as shown in Figure 4(d)). The toggling case is quite expensive in terms of power dissipation. In the next section, we discuss how a reduction in toggling events can lead to a sharp reduction in the average power dissipation on the bus lines.

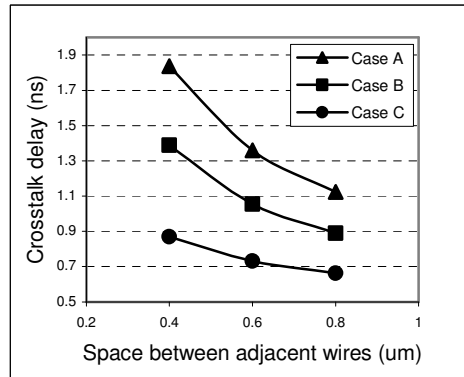


Figure 3: Crosstalk delay versus spacing between adjacent lines.

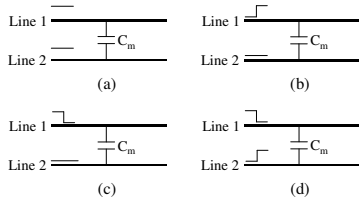


Figure 4: Coupling transitions on two adjacent lines. (a) No transition (b) Charging transitions (c) Discharging transitions (d) Toggling

## 5. Research objectives

Finding an effective solution to the aforementioned problems that are associated with chip interconnect for general application processors is crucial for current VLSI technology. To alleviate these problems we need a feasible technique that can manage handling them with low overhead.

The discussion in section 3 has shown that crosstalk noise significantly increases when the number of aggressors exceeds one. In the case of crosstalk fault occurrence and with the number of aggressors exceeding one, the number of adjacent transitions will be at least three. This observation can be utilized to detect the run-time crosstalk errors, by preventing the possibility of the natural occurrence of any three adjacent transitions in the bus lines; if such an event does somehow occur, its occurrence would signal the existence of crosstalk error. As we had earlier remarked, the worst case delay happens when the three lines switch simultaneously and the center line switches in a direction opposite to its neighbors; eliminating the occurrence of three adjacent transitions reduces the worst case delay as well.

One other important objective in this work is the decrease in the power dissipation on the bus. This can be achieved by reducing the switching activity on the bus. Not only should the switching activity on the individual lines be reduced but so should the coupling activity across the adjacent lines.

In this work we propose a coding framework for general-purpose applications that alleviates power dissipation and crosstalk problems and furthermore enables concurrent error detection on the bus lines.

## 6. Encoding framework

The goal in this section is to find a feasible encoding scheme that achieves all the objectives in this work. In this section we will use the term *codewords* to refer to the values placed in the bus by the encoder, and *codebook* to refer to the mapping between the codewords and the data words. Also we use the term *prohibited patterns* to refer to the patterns that have three consecutive transitions ( $\uparrow\downarrow\downarrow$ ). Here we introduce a methodology for designing a codebook that maps the data words to prespecified patterns of transitions that improve on power dissipation and crosstalk; the approach furthermore provides for on-line detection of errors. As we mentioned before, the on-line detection of errors and the reduction in crosstalk delay can be achieved through the prevention of the occurrence of three adjacent transitions in the bus lines. To accomplish this, we need to add a few extra lines to expand the domain so that we can have a sufficient number of feasible patterns to represent all distinct combinations in the input data after we eliminate any pattern that includes the transitions ( $\uparrow\downarrow\downarrow$ ). To determine the minimum number of extra lines ( $n_{extra}$ ), we need to solve the following inequality for the smallest integer  $n_{extra}$ :

$$2^{n+n_{extra}} - NPP(n+n_{extra}) \geq 2^n$$

Table 1: Minimum number of extra bits

Data word n bits	Min. number of extra bits ( $n_{extra}$ )	Number of combinations in n bits	Number of valid combinations in ( $n+n_{extra}$ ) bits
3	1	8	13
4	1	16	24
5	1	32	44

where  $n$  is the length of the data words and  $NPP(n+n_{extra})$  is the quantity of prohibited patterns within the set of  $2^{n+n_{extra}}$  possible patterns. Table 1 shows the results of solving this inequality for  $n$  in the range of (3-5) bits.

Providing a significant level of power savings may necessitate expanding the number of extra bits. In addition to the issues of adding extra bits, one needs also to consider the feasibility of the implementation of the encoder and the decoder. From a hardware perspective, the design of a circuit to encode a large number of bits at once may result in increasing the complexity of the design and latency. This problem can be solved through partitioning the bus into sub-buses and encoding these sub-buses individually. Extensive analysis that we have conducted indicates that the best tradeoff for the size of the sub-buses, the number of extra lines and power savings is the use of sub-buses each composed of 4 bits of data and one extra bit. Each pair of adjacent encoded sub-buses then has to be shielded from their neighbors as can be seen in Figure 5(b) to avoid the possibility of the occurrence of prohibited patterns. Thus the total number of extra lines is  $\left\lfloor \frac{n}{4} \right\rfloor + \left\lfloor \frac{n \bmod 4}{2} \right\rfloor + \left\lfloor \frac{n}{8} \right\rfloor - 1$  ( $\left\lfloor \frac{n}{4} \right\rfloor + \left\lfloor \frac{n \bmod 4}{2} \right\rfloor$  lines for encoding and  $\left\lfloor \frac{n}{8} \right\rfloor - 1$  lines for shielding). The shielding lines could be composed of any combination of VDD and VSS lines; in practice such lines might be needed anyway to provide a finer power distribution, resulting essentially in a cost free shielding.

### 6.1 Mapping algorithm

The task in this section is to determine the set of patterns of transitions that can be used in the representation of the codewords. As we discussed before, such patterns should be free of the prohibited transitions and deliver power savings. The algorithm for finding these patterns of transitions can be realized as a simple search process for a set of patterns with a minimal number of transitions through the space of the available candidates.

For a sub-bus of  $k$  bits, the task consists of finding  $2^k$  patterns of transitions with minimal number of transitions. The pattern search algorithm starts by attempting to exploit the pattern of no transitions. The algorithm is then iteratively executed, exhausting in order the patterns with increasing number of transitions until we identify the complete set of the required patterns. In this way the optimal set of required patterns is identified.

For the case of input data that are independent and uniformly distributed, changing the mapping between the data words and the predefined patterns of transitions makes no difference in the results. If the frequency distribution of the input data is far from uniform, then we can utilize this by mapping the low overhead patterns of transitions to the data words that have high frequency. Figure 5(a) shows an illustrative mapping between the input data  $b_i, b_{i+1}, b_{i+2}, b_{i+3}$  and their corresponding predefined pattern of transitions in two adjacent sub-buses  $el_0, el_1, \dots, el_4$  and  $el_5, el_6, \dots, el_9$ . It can be seen that the possibility of the occurrence of prohibited transitions is eliminated within the subgroups and across their boundaries.

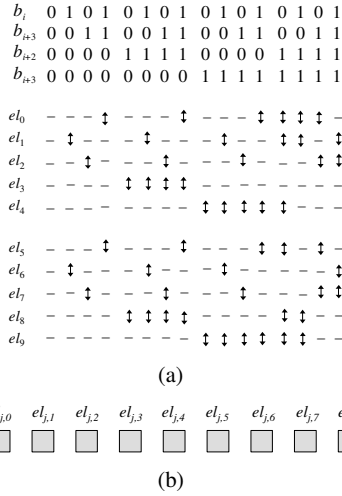


Figure 5: (a) Coding scheme. (b) Bus organization.

## 6.2 Hardware structure

Figure 6 shows the structure of the encoder and the decoder for data words of 8 bits. In the encoder side we have used JK Flip-Flops to generate the transitions. Since the mapping between the input data and the patterns of transitions is predefined, only a few gates are needed between the input data and the JK Flip-Flops. At the decoder side we have used the combination of a D-Flip-Flop and an XOR gate to transfer the transition and the steady event to logic one and zero, respectively. The task of retaining the actual data can be accomplished through a simple combinational logic. The structure of the error detection unit is shown in Figure 6. In this unit a single 3-input AND gate is used for each line in the input data to detect the occurrence of prohibited patterns.

## 6.3 Reducing self and coupling transitions

To determine how much power can be saved with the proposed coding we need to calculate the reduction in both the self and coupling transitions as power dissipation is proportional to both of them as shown in equation 2. In this section, we show an analysis under the assumption that the data are independent and uniformly distributed. In the calculations of self transitions only two events should be considered, the transition and the no transition event. The reduction in the self transitions,  $R$ , can be computed as follows:

$$\%R = \left( 1 - \frac{1}{2n} \left( \frac{13}{2} \left\lfloor \frac{n}{4} \right\rfloor - \frac{1}{4}k + 2m - \left\lfloor \frac{m}{2} \right\rfloor \right) \right) \cdot 100 \quad (3)$$

where  $m = n \bmod 4$  and  $k = \left( \left\lfloor \frac{n}{4} \right\rfloor \bmod 2 \right) \left( 1 - \left\lfloor \frac{m}{3} \right\rfloor \right)$ . For the case of  $n/4$  being an even number, the reduction in self transitions equals 18.75%.

The calculations of the coupling transitions are slightly more complex than the self transitions. Here we need to determine the number of coupling transitions in one cycle,  $n_c$ , for the three possible events between two adjacent lines ( $\rightarrow$ ), ( $\rightarrow\downarrow$ ), ( $\uparrow\downarrow$ ). For the case of the no transition event in both lines ( $\rightarrow$ ), the value of  $n_c$  is zero. If one line is quiescent and the other switches ( $\rightarrow\downarrow$ ), then  $n_c$  assumes the value of 1. To determine the value of  $n_c$  for the case of two simultaneous transitions in the two adjacent lines ( $\uparrow\downarrow$ ), we need to consider two cases.

If the two lines switch in the same direction,  $n_c$  should be 0 but if the two lines switch in opposite directions,  $n_c$  can be computed to

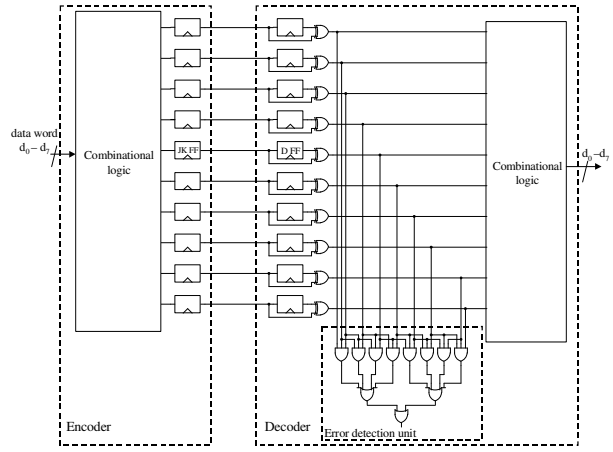


Figure 6: Encoder and decoder structure

be 4. Since the probability of these two cases is equal, the average value of  $n_c$  turns out to equal 2. The following equation shows the reduction in the coupling in a bus of  $n$  bits, where  $n$  is a multiple of 8.

$$\% \text{Reduction in coupling transitions} = \frac{3n-4}{16(n-1)} \cdot 100 \quad (4)$$

For  $n=8$  and  $n=32$ , the reduction in coupling transitions is 17.85% and 18.54%, respectively.

## 6.4 Reducing the probability of patterns that cause crosstalk noise

In this section we determine the reduction in the occurrence of patterns that cause crosstalk. Considering the mapping algorithm and the mapping example in section 6.1, one can observe that the maximum number of aggressors in the proposed coding scheme is three and can occur only across the boundaries of the adjacent sub-buses. Using the results in figure 2, it can be seen that the crosstalk noise that is associated with the three aggressors constitutes the worst case crosstalk noise in the proposed coding scheme. The results in section 3 also indicate that crosstalk errors in the proposed scheme are most likely to occur when the number of aggressors is two or three. Reducing these patterns that generate two and three aggressors is essential to reducing the possibility of errors and the overhead of the recovery process. Figures 7 and 8 show the probability of the occurrence of the two aggressors case and the three aggressors case, respectively. These results show that using the proposed coding scheme delivers a significant reduction in the probability of occurrence of the two aggressors case and a drastic reduction in the three aggressors case.

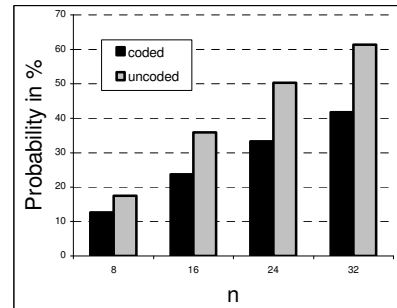


Figure 7: Two aggressor occurrence probability vs data bit number

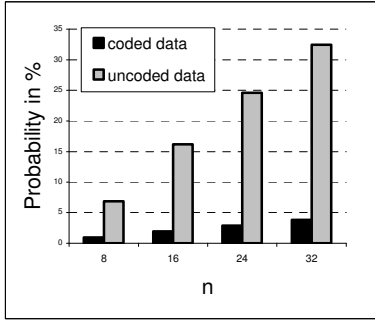


Figure 8: Three aggressor occurrence probability vs data bit number

### 6.5 Reduction in worst case crosstalk delay and noise using the proposed encoding technique.

Reducing the worst case delay and noise in the proposed coding is predicated on the elimination of prohibited patterns. The worst case crosstalk delay of the proposed technique occurs when the center line switches simultaneously in a direction opposite to one of its neighbors while the remaining neighbor is in steady case. According to the analysis that we have conducted in section 2, the worst case delay of the proposed coding scheme should be lower than the worst case crosstalk delay when no encoding is employed.

The results conducted in section 3 show that the crosstalk delay could be reduced significantly by using the proposed coding scheme. The results show also that an improved reduction in the worst case crosstalk delay could be achieved if the spacing between the adjacent lines is reduced. A reduction of 24.3% is achieved for a spacing of 0.4  $\mu\text{m}$ .

The discussion in section 6.4 has shown that the worst case crosstalk noise of the proposed coding scheme occurs when the number of aggressors is three. Using the results in Figure 2 we can observe that the worst case crosstalk noise could be reduced using the proposed coding scheme. The results show also that an improved reduction in the worst case crosstalk noise could be achieved if the spacing between the adjacent lines is reduced. A reduction of at least 10% is achieved for a spacing of 0.4  $\mu\text{m}$ .

### 7. Power savings through proposed coding scheme

The coding scheme we present is designed to reduce the average number of self and coupling transitions, which are key to the savings in power dissipation. Figure 9 shows the normalized reduction in effective transitions (self and coupling) for 8 and 32 bit bus lines with  $\frac{C_m}{C_0} = \frac{b}{1-b}$ , for data that are independent and uniformly distributed.

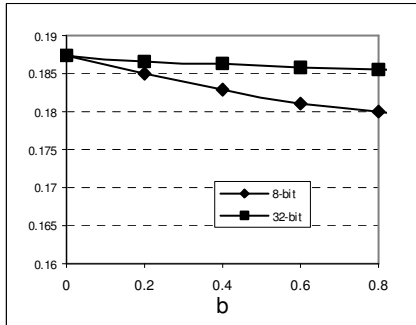


Figure 9: Normalized effective transitions for 8 and 32 bit buses.

Table 2: Transition reduction results

Benchmark	Reduction in self transitions(%)	Reduction in coupling transitions(%)
jpeg	27.60	23.49
mpeg 2	44.32	7.61
g721	39.77	17.17
gsm	31.15	25.50
epic	27.39	8.19
Avg. reduction	34.05	16.39

We have utilized the SimpleScalar toolset [12], which supports a MIPS-like instruction set architecture. The results have been obtained through the use of SimpleScalar to generate traces for instruction transfers between the instruction cache and the processor for the applications (1,000,000 transfers per application). We have used these traces to determine the effectiveness of the proposed technique to reduce the bus switching activity. The following application benchmarks were used: *image compression (jpeg)*, *video compression (mpeg2)*, *sound compression (g721)*, *sound compression (gsm)*, and *image processing (epic)*. Table 2 shows the results in the self and coupling transitions for the five benchmarks. The results show a significant reduction in both self and coupling transitions.

### 7.1 Overhead analysis

The energy overhead that is associated with the encoder and decoder is related to the number of gates that are needed to perform the encoding and decoding process. In the proposed coding scheme, the average number of gates that are needed for both encoding and decoding per bit line is close to 9 gates. For a 0.25- $\mu\text{m}$  CMOS technology, and average load capacitance of 15fF/gate [13], assuming a fan-out of 4, the energy dissipation for a bus of  $n$  lines with a 2.5V supply and  $\alpha_s = 0.5$  amounts approximately to  $0.21n$  pJ. This evaluation presents a pessimistic perspective since not all gates should be expected to switch simultaneously. Even this worst case energy overhead estimate can be considered insignificant compared to the energy dissipated in the bus lines<sup>2</sup>, as such an overhead amounts approximately to 2.8% under the conservative assumption of bus lines with  $\lambda = 1.44$ , as shown in Appendix A, and length of 15mm.

### 8. On-line detection with recovery

In this work the reliability issue is addressed by providing capabilities of on-line detection coupled with a recovery approach. By using the proposed coding scheme, the crosstalk errors in the victim line as shown in Figure 2 are most likely to occur when the number of aggressors are two or three and less likely to occur in the case of a single aggressor. As we have previously discussed, the proposed scheme is designed to prevent any simultaneous switching in any group of three adjacent lines. This condition is violated if the glitch in the victim line creates a false transition due to the effect of two or three aggressors. Such a fault can be captured through the error detection unit, shown in Figure 6, with subsequent recovery initiated through data retransmission. In addition to the presented feature of detection error at run time, the proposed scheme can also contribute in reducing the probability of occurrence of the errors caused by two and three aggressors,

<sup>2</sup> The bus line capacitance that we use in the overhead analysis is close to 3pF. As reported in [17], the average bus line capacitance can grow up to 30-50 pF, which significantly diminishes the impact of the introduced overhead even further.

Table 3: Probabilities and reduction in occurrence of two and three aggressors

Application	P(two-aggr.) (coded data) %	P(two-aggr.) (uncoded data) %	Reduction in (two-aggr) %	P(three-aggr.) (coded data) %	P(three-aggr.) (uncoded data) %	Reduction in (three-aggr) %
gsm	8.46	21.46	60.57	0.32	10.90	97.06%
mpeg	6.75	26.68	74.70	0.36	14.52	97.52%
g721	6.45	19.90	67.58	0.27	13.60	98.01%
jpeg	8.13	21.02	61.32	0.20	6.17	96.75%
epic	14.70	35.80	58.93	1.31	17.92	92.68%

as shown in Figures 7 and 8. Such reductions in likely errors benefit the recovery process, because the overhead of the retransmissions is reduced as well. Table 3 presents the simulation results for the probabilities and reduction in occurrence of two and three aggressors for various applications. The results of these simulations show a dramatic reduction in the probability of occurrence of three-aggressors and a significant reduction in the occurrence of two-aggressors.

## 9. Conclusions

In this work we have presented a coding framework for general-purpose applications that can alleviate the power dissipation and crosstalk problems on the bus lines while simultaneously detecting errors at run time. The analysis has shown that eliminating the possibility of occurrence of three consecutive transitions in three adjacent lines and reducing the infeasible patterns in terms of crosstalk noise and power dissipation can achieve the objectives of this work. In the proposed coding scheme, a predefined pattern of transitions, one for each possible combination of input data, is used to generate the codewords. An algorithm for extracting the optimized set of predefined transitions is presented. The restriction of the universe of transition patterns to the prespecified ones enables fast encoding and low hardware overhead. SPICE simulations are used to confirm the efficacy of the proposed encoding scheme in alleviating the crosstalk problem. Experimental results for various applications strongly confirm the effectiveness of using this encoding scheme to reduce power dissipation and crosstalk errors. Use of the proposed scheme promises to deliver significant benefits to current and future technologies because of the growing importance of the crosstalk problem. Such significant benefits stem from the ability to tackle at insignificant cost a significant set of the most prominent problems that are associated with deep-submicron technology.

## References

[1] M. R. Stan and W. P. Bureson, "Bus-invert coding for low-power I/O," *IEEE TVLSI*, pp. 49-58, 1995.

[2] Y. Zhang, J. Lach, K. Skadron, and M. R. Stan, "Odd/Even bus invert with two-phase transfer for buses with coupling," in *ISLPED*, pp. 80-83, 2002.

[3] P. P. Sotiriadis and A. Chandrakasan, "Low power bus encoding techniques considering inter-wire capacitance", in *CICC*, pp. 507-510, 2000.

[4] G.S. Yee, R. Christopherson, T. Throp, B.P. Wong, and C. Schen, "An automated shielding algorithm and tool for dynamic circuits," in *ISQED*, pp. 369-374, 2000.

[5] T. Xiao and M. Sadowska, "Crosstalk reduction by transistor sizing," in *ASPDAC*, pp. 137-140, 1999.

[6] C. J. Alpert, A. Devgan, and S.T. Quay, "Buffer insertion for noise and delay optimization," *IEEE TCAD*, Vol. 18, No. 11, pp. 362-367, 1999.

[7] S. Dubey and J. Jorgensen, "Crosstalk reduction using buffer insertion," *IEEE International Symposium on Electromagnetic Compatibility*, pp. 639-642, 2002.

[8] M. Lajolo, "Bus Guardians: An effective solution for online detection and correction of faults affecting system-on-chip buses," *IEEE TVLSI*, Vol. 9, No. 6, pp. 974-982, 2001.

[9] C. Metra, M. Favalli, and B. Ricco, "Self-checking detection and diagnosis of transient, delay, and crosstalk faults affecting bus lines," *IEEE TComp*, Vol. 49, No. 6, pp. 560-574, 2000.

[10] R. Ayoub, P. Petrov and A. Orailoglu, "Application specific instruction memory transformations for power efficient, fault resilient embedded processors", *IEEE International SOC Conference*, pp. 195-198, 2004.

[11] P. Petrov and A. Orailoglu, "Transforming binary code for low-power embedded processors", *IEEE Micro*, Vol. 24, No. 3, pp. 21-33, 2004.

[12] T. Austin, E. Larson and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling", *IEEE Computer*, vol. 35, n.2, pp. 59-67, 2002.

[13] J. Rabaey, A. Chandrakasan, and B. Nikolic, **Digital Integrated Circuits, A Design Perspective**, Prentice Hall, 2003.

[14] A. K. Goel, **High-speed VLSI Interconnections: modeling, analysis and simulation**, John Wiley & Sons, 1994.

[15] G. Servel, D. Deschacht, F. Saliou, J. L. Mattei, and F. Huret, "Impact of Low-K on Crosstalk," *ISQED*, pp. 298-303, 2002.

[16] D. Deschacht and G. Servel, "On-chip interconnections: Impact of adjacent lines on timing", in *ASPDAC*, pp. 539-544, 2001.

[17] S. Ramprasad, N. R. Shanbhag and I.N. Hajj, "A coding framework for low-power address and data busses", *IEEE TVLSI*, Vol. 7, No. 2, pp. 212-221, 1999

## Appendix A SPICE Configurations

A 10-stage RC ladder circuit is used to represent the interconnect line [14]. The results are evaluated using B<sup>2</sup> Spice A/D v4 pro. Table A presents the various configurations used for SPICE simulation, which are similar to the configurations that are used in [15]. S denotes the spacing between two wires, W the width of the wire, T, the thickness, H the oxide height, L the length of the wires, and W<sub>n</sub> and W<sub>p</sub> denote the widths of the N and P channels of the driver transistors, respectively. W<sub>n</sub> and W<sub>p</sub> are used in the calculations of the driver's resistances as shown in [16]. The oxide permittivity ( $\epsilon_r$ ) is 3.9, the lines are in Al-Cu with  $\rho = 3.03 \times 10^8 \Omega \cdot m$ , and the VDD is 2.5 Volt.

Table A: SPICE simulation configuration

Cases	S (um)	W (um)	T (um)	H (um)	L (um)	W <sub>n</sub> (um)	W <sub>p</sub> (um)	$\lambda(C_m/C_0)$
1	0.4	0.8	1	1	10	20	40	1.44
2	0.6	0.8	1	1	10	20	40	0.80
3	0.8	0.8	1	1	10	20	40	0.53