Exploiting Crosstalk to Speed up On-chip Buses

Chunjie Duan (c.duan@ericsson.com) *
Sunil P. Khatri (spkhatri@colorado.edu) †

Abstract

In modern VLSI processes, the cross-coupling capacitance between adjacent neighboring wires on the same metal layer is a very large fraction of the total wire capacitance. This leads to problems of delay variation due to crosstalk and reduced noise immunity, arguably one of the biggest obstacles in the design of ICs in recent times.

This problem is particularly severe in long on-chip buses, since bus signals are routed at minimum pitch for long distances. In this work, we propose to solve this problem by the use of crosstalk canceling CODECs. We only utilize memoryless CODECs, to reduce the logical complexity and enhance the robustness of our techniques.

Bus data patterns can be classified (as $4 \cdot C$, $3 \cdot C$, $2 \cdot C$, $1 \cdot C$ *or* $0 \cdot C$ *patterns) based on the maximum amount of crosstalk that they can exhibit. Crosstalk avoidance CODECs which eliminate* $4 \cdot C$ *and* $3 \cdot C$ *patterns have been reported. In this paper, we describe crosstalk avoidance techniques which eliminate* $2 \cdot C$ *and* $1 \cdot C$ *patterns. We describe an analytical methodology to accurately characterize the bus area overhead* $2 \cdot C$ *pattern CODECs. Using these results, we characterize the area overhead versus crosstalk immunity achieved. A similar exercise is performed for* $1 \cdot C$ *patterns.*

Our experimental results show that by using $2 \cdot C$ *crosstalk canceling techniques, buses can be sped up by up to a factor of 6 with an area overhead of about 200%, and that* $1 \cdot C$ *techniques are not very robust.*

1. Introduction

Crosstalk has become a significant problem in deep submicron (DSM) VLSI design. Consider the situation where we scale existing process dimensions by a factor S to obtain a new process. If this scaling is performed in all three dimensions, the cross-sectional area of wires in the design would decrease by a factor S^2. This would result in a quadratic increase in wire resistance, leading to increased wiring delays. To alleviate this problem, recent processes have avoided scaling the vertical dimension of wires (thus creating "tall" wires). This in turn has led to a situation where the cross-coupling capacitance of adjacent wires on the same layer (C_W) is much larger than the capacitance of any wire to the substrate (C_S). The ratio $r = C_W/C_S$ was shown in [5] to be around 10 for a 0.1μm process.

As a result of the large value of r, crosstalk between adjacent wires on the same metal layers manifests in ways that make designs unpredictable. It results in a significant delay variation and possible integrity problems. As a result, crosstalk has become a critical design consideration. The detrimental effects of crosstalk are aggravated in long on-chip buses, since bus signals are typically driven at minimum pitch for long distances.

The focus of this paper is the selective reduction of the delay variation effect in buses (and therefore the reduction in maximum delay and signal integrity problems in the bus signals as well) due to crosstalk. We do this by developing design techniques that allow the designer to trade-off the degree of crosstalk control desired with the associated area overhead.

The problem of delay variation of a wire due to crosstalk between adjacent wires is discussed next. Consider a group of three wires in an on-chip bus, which are driven by signals b_{i-1}, b_i and b_{i+1}. The total effective (switched) capacitance of driver b_i is dependent on the state of b_{i-1} and b_{i+1}. In the best case[1], the total effective capacitance of b_i is $C_{min} = C_S$, and in the worst case[2], the effective capacitance is $C_{max} = 4 \times C_W + C_S$. With r=10, we observe that $C_{min} = C_S$ and $C_{max} = 41 \times C_S$. This demonstrates that the delay of bus signals strongly depends on the data pattern being transmitted on the bus. For long buses, this delay variation is quite significant, as we will quantify in the sequel. As a result of this large delay variation, the worst case delay of a signal in an on-chip bus is a quantity greater than the largest possible delay of the signal, resulting in a re-

* Ericsson Wireless Communications, 6210 Spine Road, Boulder, CO 80301.

† Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80301

1 In the best case, b_{i-1}, b_i, b_{i+1} all simultaneously transition in the same direction.

2 In the worst case, b_{i-1} and b_{i+1} simultaneously transition in the opposite direction as b_i.

Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04)
1530-1591/04 $20.00 © 2004 **IEEE**

duced system performance.

This paper is organized as follows. Section 2, provides definitions used in the rest of the paper. This section also provides a classification (similar to that of [2]) of bus data patterns based on the maximum amount of crosstalk incurred by such patterns. In Section 3, we discuss previously published approaches to solving this problem. In Section 4, we describe techniques to eliminate $2 \cdot C$ crosstalk in buses. These techniques ensure that $C_{max} \leq 1 \cdot C_W$, by increasing the bus width and only using those bus vectors which guarantee that bus data pattern sequences with $C_{max} \geq 2 \cdot C_W$ can never occur. We describe a inductive approach to quantifying the increase in bus width required to achieve this type of bus immunity, using memoryless CODECs. In Section 5, we describe circuit design techniques which eliminate $1 \cdot C$ crosstalk patterns. In Section 6 we report the results of experiments that we have performed to quantify the tradeoff between the degree of crosstalk immunity achieved by the above techniques, and the bus area overhead incurred. We compare our results with those reported in [2], in which we described memoryless CODECs to eliminate $4 \cdot C$ and $3 \cdot C$ crosstalk patterns. We conclude the paper in Section 7.

2. Preliminaries

In this section, we introduce the classification scheme for bus data transitions which we will utilize in the sequel. Our classification is largely borrowed from that introduced in [2].

Consider an n-bit bus, consisting of signals $b_1, b_2, b_3 \cdots b_{n-1}, b_n$.

Definition 1 *: A* **Vector** *v is an assignment to the signals b_i as follows:*
$b_i = v_i$, (where $1 \leq i \leq n$ and $v_i \in \{0,1\}$).

Definition 2 *: The* **Complement of a Vector** *v (denoted by $\overline{v}$) is a vector for which the signals b_i are assigned values:*
$b_i = \overline{v_i}$, (where $1 \leq i \leq n$ and $v_i \in \{0,1\}$).

Consider two successive vectors v_j and v_{j+1}, being transmitted on a bus. For vector v_j, assume $b_i = v_i^{v_j}$ (where $1 \leq i \leq n$ and $v_i^{v_j} \in \{0,1\}$). Similarly, for vector v_{j+1}, assume $b_i = v_i^{v_{j+1}}$ (where $1 \leq i \leq n$ and $v_i^{v_{j+1}} \in \{0,1\}$).

Consider vector sequence $v_1, v_2, \cdots, v_j, v_{j+1}, \cdots v_k$, applied on a bus. We define five types of crosstalk conditions below. For these definitions, we assume that $0 \leq i \leq n-2$ and $0 \leq j \leq k-1$.

Definition 3 *A sequence of vectors is called a* **4·C sequence** *if $\exists i, j$ s.t.*
$v_i^{v_j} = v_{i+1}^{v_{j+1}} = v_{i+2}^{v_j} = v$ and $v_i^{v_{j+1}} = v_{i+1}^{v_j} = v_{i+2}^{v_{j+1}} = \overline{v}$, where $v \in \{0,1\}$.

Definition 4 *A sequence of vectors is called a* **3·C sequence** *if it is not a 4·C sequence and $\exists i, j$ s.t.*

- *$v_i^{v_j} = v_{i+1}^{v_{j+1}} = v_1$ and $v_i^{v_{j+1}} = v_{i+1}^{v_j} = \overline{v_1}$ and $v_{i+2}^{v_j} = v_{i+2}^{v_{j+1}} = v_2$ where $v_1, v_2 \in \{0,1\}$ OR*
- *$v_{i+1}^{v_j} = v_{i+2}^{v_{j+1}} = v_1$ and $v_{i+1}^{v_{j+1}} = v_{i+2}^{v_j} = \overline{v_1}$ and $v_i^{v_j} = v_i^{v_{j+1}} = v_2$ where $v_1, v_2 \in \{0,1\}$.*

Definition 5 *A sequence of vectors is called a* **2·C sequence** *if it is not a 4·C or 3·C sequence and $\exists i, j$ s.t.*
$v_i^{v_j} = v_i^{v_{j+1}} = v_1$ and $v_{i+1}^{v_j} = v_2$ and $v_{i+1}^{v_{j+1}} = \overline{v_2}$ and $v_{i+2}^{v_j} = v_{i+2}^{v_{j+1}} = v_3$, where $v_1, v_2, v_3 \in \{0,1\}$.

Definition 6 *A sequence of vectors is called a* **1·C sequence** *if it is not a 4·C, 3·C or 2·C sequence and $\exists i, j$ s.t.*

- *$v_i^{v_j} = v_i^{v_{j+1}} = v_1$ and $v_{i+1}^{v_j} = v_{i+2}^{v_j} = v_2$ and $v_{i+1}^{v_{j+1}} = v_{i+2}^{v_{j+1}} = \overline{v_2}$, where $v_1, v_2, v_3 \in \{0,1\}$ OR*
- *$v_{i+2}^{v_j} = v_{i+2}^{v_{j+1}} = v_1$ and $v_i^{v_j} = v_{i+1}^{v_j} = v_2$ and $v_i^{v_{j+1}} = v_{i+1}^{v_{j+1}} = \overline{v_2}$, where $v_1, v_2, v_3 \in \{0,1\}$.*

Definition 7 *A sequence of vectors is called a* **0·C sequence** *if it is not a 4·C, 3·C, 2·C, or 1·C sequence.*

Definition 8 *An set C_n of n bit vectors is said to be a $k \cdot C$* **crosstalk free clique** *iff any vector sequence made up of vectors $v \in C_n$ is a $l \cdot C$ sequence, where $l < k$.*

If a sequence of vectors on a bus is a k·C sequence ($0 \leq k \leq 4$), then the physical interpretation of this is that:

- This vector sequence has at least one bit b for which there exists consecutive vectors that require the driver of this bit to charge a capacitance $k \cdot C_W + C_S$. Note that $C_W \gg C_S$.
- This vector sequence has no bit for which there exists consecutive vectors that require the driver of this bit to charge a capacitance greater than $k \cdot C_W + C_S$.

From the above, it is clear that as k increases, the corresponding bus signal b is increasingly slower. Further, it would be desirable to devise encoders which ensure that an arbitrary vector sequence is converted into a $k \cdot C$ vector sequence which is transmitted over the bus, and then decoded again at the receiving end. CODECs which eliminate $4 \cdot C$ and $3 \cdot C$ sequences have been described in [2]. In this paper, we discuss techniques to eliminate $2 \cdot C$ and $1 \cdot C$ sequences.

3. Previous Work

Crosstalk reduction for on-chip buses has been the focus of some recent research. In [10], the main contribution of the authors was to extend the Elmore delay model to account for distributed nature of self and cross-coupling capacitances in on-chip buses. They suggest the possibility of using CODECs to eliminate certain bus transitions. They also suggest that encoding could speed up buses by $2\times$ (this would be achieved by ensuring that bus never exhibits $4 \cdot C$ or $3 \cdot C$ transitions). In [2], we classify bus data transitions

from a crosstalk viewpoint, and describe CODECs to eliminate $4 \cdot C$ and $3 \cdot C$ transitions on the bus. They show that the asymptotic overhead when eliminating $3 \cdot C$ transitions is about 44%. The CODECs described in [2] are memoryless. The authors of [11] discuss memory based as well as memoryless encoding techniques to eliminate crosstalk. However, area and delay overheads due to CODEC implementation were not quantified. In [4], the authors reduce crosstalk induced delay variation in buses by selectively skewing bus data signals. Finally, [6] proposes a bus repeater sizing methodology which accounts for crosstalk induced delays and controls them by upsizing the drivers. This could result in driver circuits with large power and area requirements.

In [7], the authors describe a technique to simultaneously minimize bus power consumption while eliminating $3 \cdot C$ and $4 \cdot C$ crosstalk. The possibility of eliminating $1 \cdot C$ and $2 \cdot C$ crosstalk is not discussed. Further, the overhead for CODEC implementation is not discussed. The overhead in terms of bus size, of their $3 \cdot C$ crosstalk eliminating CODEC is between 62.5% and 72% (depending on bus size), in contrast to the asymptotic overhead of 44% reported in [2] and [11]. The work of [8] focuses on bus energy as opposed to delay. No CODECs are utilized, rather the approach is to adjust the spacing between bus wires non-uniformly (based on specific bus data statistics), with the ultimate goal of reducing bus energy. However, the worst case bus bit still incurs $4 \cdot C$ crosstalk, so delay is reduced (due to the increase in wire spacing) only minimally.

In [12], [3] and [13], the authors focus on routing techniques which utilize crosstalk information about the wires being routed. The work of [14] aims to reduce crosstalk in datapath circuits. In contrast to these approaches, our paper focuses on crosstalk in buses (where the problem is significantly more acute).

In this work, in contrast to [10], [2], [4], [6] and [11], we provide techniques to *speed up* a bus even further, by ensuring that the bus never exhibits $2 \cdot C$ or $1 \cdot C$ transitions.

4. Eliminating $2 \cdot C$ Crosstalk

We first devise an inductive method to construct a $2 \cdot C$ crosstalk free clique of any size k. Since it is easy to construct crosstalk immune cliques for small k, we can use our inductive method to construct arbitrary-sized crosstalk immune cliques.

Suppose we are given an n-bit $2 \cdot C$ crosstalk free clique C_n. Let our $(n+1)$-bit $2 \cdot C$ crosstalk free clique be initialized to the empty set. Since each bit b_i $(1 < i < n)$ of the bus can have at most $1 \cdot C$ crosstalk, therefore, if we duplicate the value of the n^{th} bit of every vector $v \in C_n$ into the $(n+1)^{th}$ bit position, we get legal vectors[3] of C_{n+1}. So all v' satisfying the property

$$v' = v \cdot v_n$$

are legal members of C_{n+1} (where "$\cdot$" is the concatenation operator). Starting with $C_{n+1} = \varphi$, we add vectors to it as follows:

$$C_{n+1} \leftarrow C_{n+1} \cup v'$$

This is done for each $v \in C_n$. As a result, C_{n+1} is at least as large as C_n. Also, since we duplicated the value of v_n into the $(n+1)^{th}$ position of the $(n+1)$-bit vector, b_{n+1} is guaranteed to have $0 \cdot C$ crosstalk. This ensures that C_{n+1} is a $2 \cdot C$ crosstalk free clique. Since we intend to construct a $2 \cdot C$ crosstalk free clique, we may add vectors to the existing C_{n+1} that we have, so that b_{n+1} has up to $1 \cdot C$ crosstalk.

To augment C_{n+1} in this manner, we define S_1 and S_0 (which form a partition of C_{n+1}) as follows:

Definition 9 *The* **0-extended subset of** C_{n+1} *is defined as*

$$S_0 = \{v \in C_{n+1} | v_{n+1} = 0\}$$

Definition 10 *The* **1-extended subset of** C_{n+1} *is defined as*

$$S_1 = \{v \in C_{n+1} | v_{n+1} = 1\}$$

We now construct two new sets $C_{n+1}^{S_0}$ and $C_{n+1}^{S_1}$ of candidate elements for C_{n+1} as below. These sets are constructed in a manner that $C_{n+1} \cup C_{n+1}^{S_0}$ or $C_{n+1} \cup C_{n+1}^{S_1}$ are both $2 \cdot C$ crosstalk free cliques.

Constructing $C_{n+1}^{S_0}$: For each distinct vector $v \in S_0$, we create a new vector v^{new} using the following construction:

$$v_i^{new} = v_i \qquad (0 \le i \le n)$$

and

$$v_{n+1}^{new} = \overline{v_{n+1}}$$

This vector v^{new} is appended to $C_{n+1}^{S_0}$ unless, there exists a vector $w \in S_1$, such that:

$$v_n^{new} \neq w_n$$

and

$$v_{n-1}^{new} = w_{n-1}$$

Constructing $C_{n+1}^{S_1}$: The process is similar to the construction of $C_{n+1}^{S_0}$. The only difference is that $v \in S_1$ and $w \in S_0$.

Finally,

$$C_{n+1} \leftarrow C_{n+1} \cup C_{n+1}^{S_p}$$

where

$$p = \underset{i}{argmax}(|C_{n+1}^{S_i}|)$$

In other words, we append the larger of the sets $C_{n+1}^{S_0}$ or $C_{n+1}^{S_1}$ to C_{n+1}.

3 in the sense of not having more than $1 \cdot C$ crosstalk sequences among themselves

4.1. Proof of Correctness

We state and prove a theorem about the correctness of the above steps in inductively constructing a $(n+1)$-bit $2 \cdot C$ crosstalk free clique

Theorem 4.1 *The sets $C_{n+1} \cup C_{n+1}^{S_0}$ and $C_{n+1} \cup C_{n+1}^{S_1}$ are both $2 \cdot C$ crosstalk free cliques.*

Proof: We have already shown that C_{n+1} is a $2 \cdot C$ crosstalk free clique. Now consider the construction of $C_{n+1}^{S_0}$. Since $v_i^{new} = v_i$ and $v_{n+1}^{new} = \overline{v_{n+1}}$ for a given $v \in S_0$, the resulting vector v^{new} has no more than $1 \cdot C$ crosstalk with any $v \in S_0$.

To prove that v^{new} has no more than $1 \cdot C$ crosstalk with any $w \in S_1$, the argument proceeds as follows. Since $v_{n+1}^{new} = w_{n+1}$ (by construction), b_n of v^{new} has no more than $1 \cdot C$ crosstalk with b_{n+1} of any $w \in S_1$. Since we prune all v^{new} such that $v_n^{new} \neq w_n$ and $v_{n-1}^{new} = w_{n-1}$ (for any w), we ensure that b_n of v^{new} has no more than $1 \cdot C$ crosstalk with b_{n-1} of any $w \in S_1$. Therefore the resulting set $C_{n+1} \cup C_{n+1}^{S_0}$ is a $2 \cdot C$ crosstalk free clique ■

$C_{n+1} \cup C_{n+1}^{S_1}$ can be shown to be a $2 \cdot C$ crosstalk free clique in a similar manner.

4.2. Efficient inductive construction of $(n+1)$-bit $2 \cdot C$ crosstalk free cliques

We could use the construction above to generate $(n+1)$-bit $2 \cdot C$ crosstalk free cliques. However, this construction may involve significant pruning operations. As a result, in practice, we utilize a more efficient inductive construction which eliminates the need to prune vectors.

Let T_{xy}^n be the number of vectors $v \in C_n$ which satisfy $(v_{n-1} = x) \wedge (v_n = y)$, where $x, y \in 0, 1$. Therefore, $T_{00}^n + T_{01}^n + T_{10}^n + T_{11}^n = |C_n|$. Note that one of T_{01}^n or T_{10}^n is 0 trivially (otherwise C_n cannot be a $2 \cdot C$ crosstalk free clique).

Now we define a set of recurrence equations to construct C_{n+1} from C_n. We construct two variants of C_{n+1} (using both the 0-extended subset and the 1-extended subset), and finally select the larger of the two.

4.3. Constructing C_{n+1} from C_n using the 0-extended subset

The algorithm below describes our inductive algorithm.

Note that the first two lines of this algorithm simply duplicate the last bit of each vector in C_n to create C_{n+1}. Note that since we are using the 0-extended subset to construct C_{n+1}, $T_{10}^{n+1} = 0$ as indicated in the third line. Lines 4-9 are explained as follows. There can be as many T_{01}^{n+1} vectors as there are either T_{10}^n or T_{00}^n vectors. In the first case, we require that $T_{11}^n = 0$ to avoid any $2 \cdot C$ sequences. In the second case, we require $T_{01}^n = 0$ to avoid any $2 \cdot C$ sequences.

Algorithm 1 Constructing C_{n+1} from C_n using the 0-extended subset

$T_{00}^{n+1} = T_{00}^n + T_{10}^n$
$T_{11}^{n+1} = T_{01}^n + T_{11}^n$
$T_{10}^{n+1} = 0.$
if $(T_{01}^n = 0)$ **then**
  $T_{01}^{n+1} += T_{00}^n$
end if
if $(T_{11}^n = 0)$ **then**
  $T_{01}^{n+1} += T_{10}^n$
end if

4.4. Constructing C_{n+1} from C_n using the 1-extended subset

The algorithm is similar to the previous one, with minor changes in lines 4 through 9. For the sake of brevity, we have omitted the explanation of these lines, since it is similar to the explanation above.

In practice, we ensure that the C_{n+1} which we construct is maximal, by using as a starting point all possible cliques C_n.

4.5. Area Overhead Trends

The results obtained using the above method for constructing $2 \cdot C$ crosstalk free cliques are shown in Figure 1. The figure describes the relationship between the actual bus size (y-axis) and the effective bus size (x-axis) which is given by $m = \lfloor log_2(C_n) \rfloor$. This is plotted using a dotted line. The percentage overhead is shown as well (solid line). We note that the asymptotic overhead is about 146%.

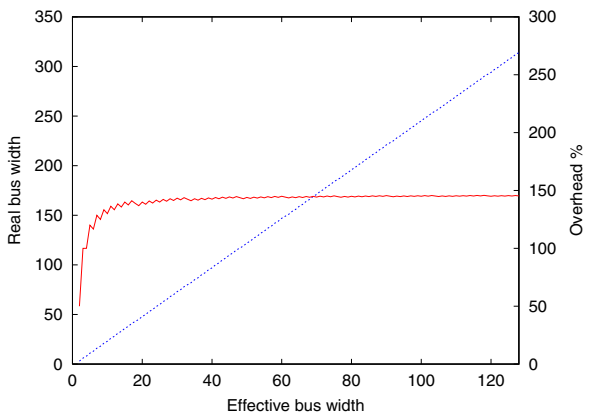


Figure 1. Effective versus actual bus width

5. Eliminating $1 \cdot C$ Crosstalk

We explore several alternatives to eliminating $1 \cdot C$ crosstalk, with the basic idea being that we surround a bus wire b_i by other wires transmitting identical information, such that the central b_i wire incurs $0 \cdot C$ crosstalk

trc_length	bufsize	0c	1c	2c	3c	4c
5mm	30×	83	121	241	516	665
5mm	60×	108	131	213	399	402
5mm	120×	96	117	136	196	279
10mm	30×	102	153	437	912	1026
10mm	60×	131	164	413	722	919
10mm	120×	114	137	270	379	548
20mm	30×	153	203	793	1068	1586
20mm	60×	164	206	691	1161	1561
20mm	120×	134	177	580	969	1365

Table 1. Delay comparison for different driver size and trace length (ps)

(since its neighbors always transition in the same direction). Let us call the left, center and right wires of bit b_i as b_i^l, b_i^c and b_i^r respectively. Note that in this scenario, no CODEC is required, and this scheme has an overhead of at least 200%.

We compare 5 different configurations to eliminate $1 \cdot C$ crosstalk. These configurations are described in Figure 2.

- Configuration a) consists of groups of 3 minimum width wires for each bus signal, separated by minimum spacing. Groups of these wires are separated by a variable distance.
- Configuration b) is similar to a) with the exception of the fact that groups of bus signal wires are separated by a GND signal.
- Configuration c) consists of a single wire (of varying width) for each bus signal, with adjacent bus signal wires separated by a variable spacing.
- Configuration d) is similar to a) with the difference that each bus signal consists of a group of 5 minimum width wires separated by minimum spacing. Groups of these wires are separated by a variable distance.

In configurations a), b) and d), the signal of the central bus signal wire is used by the receiving circuitry.

6. Experimental Results

Table 1 reports the worst-case delay among the bus signals under all cross-talk conditions. The results were generated using SPICE [9]. A 0.1μm process was used, and buses were assumed to be routed on Metal4. Wiring parasitics were obtained from [1] using the interconnect predictions reported in [5]. Wires were modeled as distributed RC transmission lines. In Table 1, the first column reports the length of the bus wires. Column 2 reports the driver size (in multiples of a minimum-sized driver). Columns 3 through 7 report the worst case delay of the bus in picoseconds, assuming that no greater than 0·C through 4·C cross-talk patterns are allowed respectively.

From the above, we can note that eliminating $2 \cdot C$ or $1 \cdot C$ transitions on a bus can speed up the bus significantly. For example, in the configuration of 60× drivers, 10mm wires, the delay reduces by about 6× compared to the $4 \cdot C$ case.

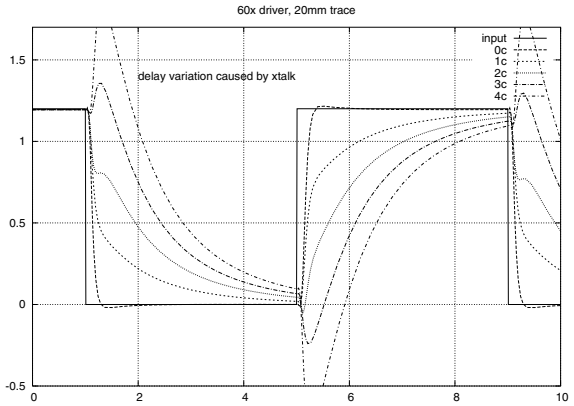


Figure 3. Sample SPICE Waveforms (60× driver, 20mm trace length, 0.1μ process)

A SPICE plot for the delays (in nanoseconds) of signals representing each of the 5 classes of cross-talk is shown in Figure 3.

6.1. $2 \cdot C$ Experiments

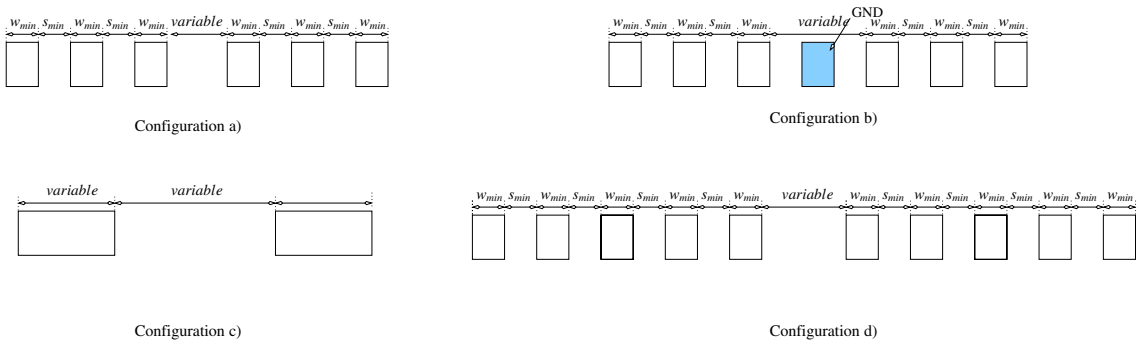
We implemented our $2 \cdot C$ crosstalk free CODEC using PLAs. In particular, we implemented a 3→6 bit encoder, and determined its delay to be 179ps, and its area to be 10 μm^2. For larger buses, we would perform the encoding by concatenating an appropriate number of such encoders. When adjacent bits of adjacent encoded buses have an opposite value, we complement the encoded vector of one of these buses, avoiding $2 \cdot C$ crosstalk between encoded buses.

It is worth mentioning that in this case, the CODEC speed can be the limiting factor on system performance, since many entries under the column $1 \cdot C$ of Table 1 report a delay less than the encoder delay. In spite of this, the speedup of the data transfer is quite dramatic (approximately between 3× and 7×).

6.2. $1 \cdot C$ Experiments

Figure 4 compares the schemes of Figure 2. The delays in this figure correspond to neighboring groups of 3 wires (which represent the same bus signal) switch in opposite directions (resulting in a maximum delay). The length of wires was 20mm, and the driver was a 30× minimum.

Configuration c) resulted in significantly larger delay penalties that the other configurations, because of the absence of any shielding. Among the other configurations, we note that the minimum delay is greater than the delay of $0 \cdot C$ signals from Table 1. This is because it is exceedingly hard to ensure that all 3 (or 5) wires (representing the same bus signal) switch at *exactly* the same time (which is required



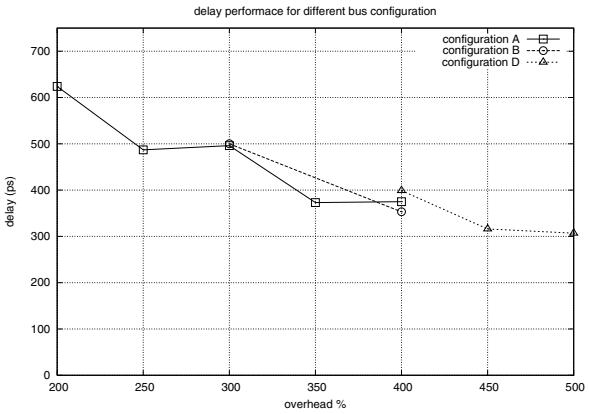


Figure 4. Delay versus Area tradeoff for 1C schemes

for a $0 \cdot C$ transition). The possible skew on the outer shielding wire is sufficiently large to cause the transition of the center wire to effectively have close to $2 \cdot C$ crosstalk. This was verified by simulations where we intentionally skewed the outer and central signals.

7. Conclusions

With the decreasing feature sizes of VLSI processes, the cross-coupling capacitance of adjacent wires on the same metal layer dominates the capacitance of any wire to substrate. In this paper, we have discussed techniques to speed up a bus by exploiting the crosstalk between its wires.

We have introduced CODECs to eliminate $2 \cdot C$ crosstalk in a bus, and also described techniques to eliminate $1 \cdot C$ crosstalk. We have shown that the $2 \cdot C$ crosstalk elimination can be achieved with a speedup of up to $6\times$ and a 200% area overhead. $1 \cdot C$ crosstalk elimination was shown to be not sufficiently robust.

References

[1] Physical Design Modelling and Verification Project (SPACE Project). `http://cas.et.tudelft.nl/research/space/html`.

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

[3] T. Gao and C. Liu. Minimum crosstalk channel routing. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 692–696, Santa Clara, CA, Nov 1993.

[4] K. Hirose and H. Yasuura. A bus delay reduction technique considering crosstalk. In *Proceedings of Design, Automation and Test in Europe (DATE)*, pages 441–445, Paris, France, Mar 2000.

[5] S. Khatri, R. Brayton, and A. Sangiovanni-Vincentelli. *Cross-Talk Noise Immune VLSI design using regular layout Fabrics*. Kluwer Academic Publishers, 2001. ISBN 0-7923-7407-X.

[6] D. Li, A. Pua, P. Srivastava, and U. Ko. A repeater optimization methodology for deep sub-micron, high-performance processors. In *IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD)*, pages 726–731, Austin, TX, Oct 1997.

[7] C.-G. Lyuh and T. Kim. Low power bus encoding with crosstalk delay elimination. In *15th Annual IEEE International ASIC/SOC Conference*, pages 389–393, Sept 2002.

[8] L. Macchiarulo, E. Macii, and M. Poncino. Wire placement for crosstalk energy minimization in address buses. In *Proceedings of Design, Automation and Test in Europe (DATE)*, pages 158–162, Paris, France, Mar 2002.

[9] L. Nagel. Spice: A computer program to simulate computer circuits. In *University of California, Berkeley UCB/ERL Memo M520*, May 1995.

[10] P. Sotiriadis and A. Chandrakasan. Reducing bus delay in submicron technology using coding. In *Proceedings Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 109–114, Yokohama, Japan, Jan/Feb 2001.

[11] B. Victor and K. Kuetzer. Bus encoding to prevent crosstalk delay. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 57–63, San Jose, CA, Nov 2001.

[12] A. Vittal and M. Marek-Sadowska. Crosstalk reduction for vlsi. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(3):290–298, Mar 1997.

[13] T. Xue, E. Kuh, and D. Wang. Post global routing crosstalk synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(12):1418–1430, Dec 1997.

[14] J.-S. Yim and C.-M. Kyung. Reducing cross-coupling among interconnect wires in deep-submicron datapath design. In *Proceedings. 36th Design Automation Conference (DAC)*, pages 485–490, New Orleans, LA, Jun 1999.