Analysis and Avoidance of Cross-talk in On-Chip Buses

Chunjie Duan (c.duan@ericsson.com) *
Anup Tirumala (anup@JasmineNetworks.com) †
Sunil P. Khatri (spkhatri@colorado.edu) ‡

Abstract

We present techniques to analyze and alleviate cross-talk in on-chip buses. With rapidly shrinking process feature sizes, wire delay is becoming a large fraction of the overall delay of a circuit. Additionally, the increasing cross-coupling capacitances between wires on the same metal layer create a situation where the delay of a wire is strongly dependent on the electrical state of its neighboring wires. The delay of a wire can vary widely depending on whether its neighbors perform a like or unlike transition.

This effect is acute for long on-chip buses. In this work, we classify cross-talk interactions between the wires of an on-chip bus. We present encoding techniques which can help a designer trade off cross-talk against area overhead. Our experimental results show that the proposed techniques result in reduced delay variation due to cross-talk. As a result, the overall delay of a bus actually decreases even after the use of the encoding scheme.

1. Introduction

As VLSI fabrication technologies advance into the deep sub-micron region, the inter-wire capacitance (C_I) becomes significant compared to the wire-to-substrate capacitance (C_L). Using their "strawman" process predictions, the authors of [1] demonstrated that for a 0.1μm process, the ratio $r = C_I/C_L$ is typically 10. Since C_I is the dominant capacitance for deep sub-micron processes, cross-talk between adjacent wire becomes significant. This effect is especially important for long on-chip buses.

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 capacitance of driver b_i is dependent on the state of b_{i-1} and b_{i+1}. In the best case[1], the capacitance is $C_{min} = C_L$, and in the worst case[2], the capacitance is $C_{max} = 4 \times C_I + C_L$. With r=10, we observe that $C_{min} = C_L$ and $C_{max} = 41 \times C_L$. This shows that the delay of bus signals strongly depend on the bus data pattern.

In this paper, we first classify bus data sequences based on a measure of the cross-talk that they would incur. We show that C_{max} can be reduced by eliminating some undesirable data sequences on the bus (and by increasing the bus width). We derive mathematical bounds on the number of additional bus signals required for this purpose. We illustrate encoding techniques that trade off cross-talk tolerance against area overhead. Experimental results demonstrate that our techniques are able to reduce the worst case delay of a bus, even after including encoding overheads.

Though there has been much work (an incomplete list is [2, 3]) on encoding buses for low power, very little work has been published on encoding buses for low cross-talk.

In [2], the authors simply complement all signals of the bus whenever more than 50% of the bus signals switch. Overhead is a reasonable amount of logic, and 1 signal per bus (regardless of bus size).

The work of [3] offers a insightful formulation of the problem of bus energy minimization. However, it is not applicable in the context of the cross-talk problem, since worst-case cross-talk patterns can occur even among the patterns that have very low energy consumption

The rest of this paper is organized as follows. In Section 2, we provide definitions used in the rest of the paper. Section 3 outlines techniques to eliminate "3-C" cross-talk patterns, while Section 4 illustrates methods to eliminate "4-C" cross-talk patterns. Experimental results are presented in Section 5, and conclusions are drawn in Section 6.

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

†Jasmine Networks, 3061 Zanker Road, Suite B, San Jose, CA 95134

‡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.

0-7695-1357-3/01 $10.00 © 2001 IEEE

2. Preliminaries and Terminology

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 *: A vector* V *is a* **Forbidden Vector** *if the following condition is satisfied:*
$b_i = v$
$b_{i+1} = \overline{v}$
$b_{i+2} = v$
where $0 \leq i \leq n-2$, *and* $v \in \{0,1\}$

Definition 3 *: 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 $b_i^{V_j} = v_i^{V_j}$ (where $1 \leq i \leq n$ and $v_i^{V_j} \in \{0,1\}$. Similarly, for vector V_{j+1}, assume $b_i^{V_{j+1}} = v_i^{V_{j+1}}$ (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 cross-talk conditions below. For these definitions, we assume that $0 \leq i \leq n-2$ and $0 \leq j \leq k-1$.

Definition 4 *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 5 *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 6 *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 7 *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 8 *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.*

Intuitively, when a bus incurs a 4·C vector sequence, the transition of signal b_{i+1} is significantly slowed down due to the fact that its neighbors transition in an opposite sense. An effective capacitance of $4 \times C$ is to be charged by signal b_{i+1} in this case (assuming the capacitance between adjacent signals in the bus is C). However, for a 0·C sequence, the adjacent signals transition in a similar sense and the cross-coupling capacitance to be charged is 0. As we discussed earlier, C is a large quantity in deep sub-micron technologies, resulting in large delay variations of bus signals depending on the nature of the vector sequence applied.

3. Eliminating 3·C Cross-talk Sequences

In the following discussion, we assume that $r >> 1$ and therefore C_L has a negligible contribution to the total capacitance. For two wires b_i and b_{i+1} routed side by side, if the transitions on both are in the same direction ($v_i^{V_j} = v_{i+1}^{V_j} = v$ and $v_i^{V_{j+1}} = v_{i+1}^{V_{j+1}} = \overline{v}$ where $v \in \{0,1\}$), then the maximum effective capacitance (denoted C_T^{max}) for either wire is zero.

If a transition occurs only on one wire and the other wire has no transition ($v_i^{V_j} = v_i^{V_{j+1}} = v_1$, and $v_{i+1}^{V_j} \neq v_{i+1}^{V_{j+1}}$) then $C_T^{max} = C_I$.

The maximum effective capacitance for either wire occurs when $v_i^{V_j} = v_{i+1}^{V_{j+1}} = v$ and $v_i^{V_{j+1}} = v_{i+1}^{V_j} = \overline{v}$ where $v \in \{0,1\}$. Also, $C_T^{max} = 2 \times C_I$.

For 3 adjacent bits in any bus (b_i, b_{i+1} and b_{i+2}), the maximum capacitance on b_i is $4 \times C_I$, which happens when $\exists j$ *s.t.* vector V^j is a **forbidden vector** and $v_i^{V_j} = v_{i+1}^{V_{j+1}} = v_{i+2}^{V_j} = v$ and $v_i^{V_{j+1}} = v_{i+1}^{V_j} = v_{i+2}^{V_{j+1}} = \overline{v}$, where $v \in \{0,1\}$.

Theorem 3.1 *If forbidden vectors are not allowed on the bus,* $C_T^{max} = 2 \times C_I$.

Proof: Since forbidden vectors are excluded, it is easy to see that $C_T^{max} < 4 \times C_I$, by the argument made above. Let us assume that $C_T^{max} = 3 \times C_I$. This implies that

- $v_i^{V_j} = v_{i+1}^{V_{j+1}} = v_1$ 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 $v_1, v_2 \in \{0,1\}$.

Both cases above require the use of a forbidden vector. ■

For an n-bit bus, there are 2^n distinct vectors. For an $n+m$ bit bus, there are $T_g(n+m)$ vectors that are not forbidden vectors. If $T_g(n+m) > 2^n$, we can find an encoder that map each input (n-bit) vector uniquely to an output ($n+m$ bit) vector such that none of the output vectors is forbidden. After such an encoding, we can guarantee that the maximum inter-wire capacitance is $2 \times C_I$ or less. Without this encoding, the maximum inter-wire capacitance may be as high as $4 \times C_I$.

In order to derive such an encoder, we first need to be able to determine $T_g(n)$. This is the focus of the next section.

3.1. Counting Forbidden Vectors

Definition 9 *For an n-bit bus, we define the following quantities:*

- $T(n)$ *is the total number of distinct n-bit vectors.*
- $T_b(n)$ *is the total number of* **forbidden vectors.**
- $T_g(n)$ *is the total number of* **legal vectors** *(not forbidden).*
- $T_{gg}(n)$ *is the number of legal vectors satisfying* $v_{n-1} = v_n$.
- $T_{gb}(n)$ *is the number of legal vectors satisfying* $v_{n-1} \neq v_n$

Based on the definitions, we know that:
$T(n) = T_g(n) + T_b(n)$ and $T_g(n) = T_{gg}(n) + T_{gb}(n)$

We want to encode the vectors of an n-bit bus using an $n+m$ bit bus, such that $T_g(n+m) > 2^n$. This would ensure that all $4 \times C_I$ and $3 \times C_I$ cross-talk patterns are eliminated. We now propose an inductive scheme to compute a lower bound for the overhead involved in this process.

All possible vectors for an n-bit bus $\{V(n)\}$ can be generated from $\{V(n-1)\}$, by appending one bit to each element of $\{V(n-1)\}$. From each distinct $V(n-1)$, two distinct $V(n)$ are generated, $b_1 b_2 ... b_{n-1} 0$ and $b_1 b_2 ... b_{n-1} 1$.

- If the vector $b_1 b_2 ... b_{n-1}$ is a **forbidden vector**, both $V(n)$ vectors generated from it must be **forbidden vectors** too.
- If $V(n-1)$ is a legal vector, and $v_{n-2} \neq v_{n-1}$, then one of the $V(n)$ vectors generated will have a **forbidden pattern**. The other one will not have the forbidden pattern since its last two bits are identical ($v_{n-1} = v_n$).

The base case for the inductive argument above is shown below, for a 3-bit bus.

Base case (3-bit bus):

$T(3) = 8$;
$T_g(3) = 6$;
$T_b(3) = 2$;
$T_{gg}(3) = 4$;
$T_{gb}(3) = 2$;

Inductive step:

$T(n) = 2 \times T(n-1)$;
$T_g(n) = 2 \times T_{gg}(n-1) + T_{gb}(n-1)$;
$T_{gg}(n) = T_{gg}(n-1) + Tgb(n-1)$;
$T_{gb}(n) = T_{gg}(n-1)$;
$T_b(n) = 2 \times T_b(n-1) + T_{gb}(n-1)$;

From these equations, we can construct a simple recurrence equation for $T_g(n)$. The overhead (i.e. the number of additional bits used) percentage is computed using the following equations, and is illustrated in Figure 1:

$$m = \lfloor log_2(T_g(n)) \rfloor \quad (1)$$

$$Oh(m) = min((n-m)/m) \quad (2)$$

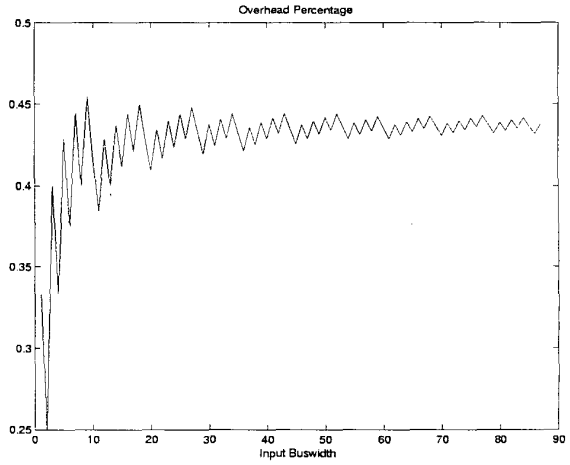

Figure 1. Percentage Overhead

3.2. Encoding Techniques

Table 1 shows a particular mapping which eliminates $3 \cdot C$ (and $4 \cdot C$) crosstalk sequences. More efficient mapping may exist. For the particular encoding shown in Table 1, the logic functions are:

$q_1 = d_1$
$q_1 = (d_2 d_3 + d_2 \overline{d_4}) \oplus d_1$
$q_3 = (d_2 + d_3 \overline{d_4}) \oplus d_1$
$q_4 = (d_3 + d_2 d_4) \oplus d_1$
$q_5 = d_4 \oplus d_1$

The encoder we implemented is shown in Figure 2, for a 16-bit bus. The input is divided into four 4-bit groups and the data of each group is encoded using a 4-5 encoder.

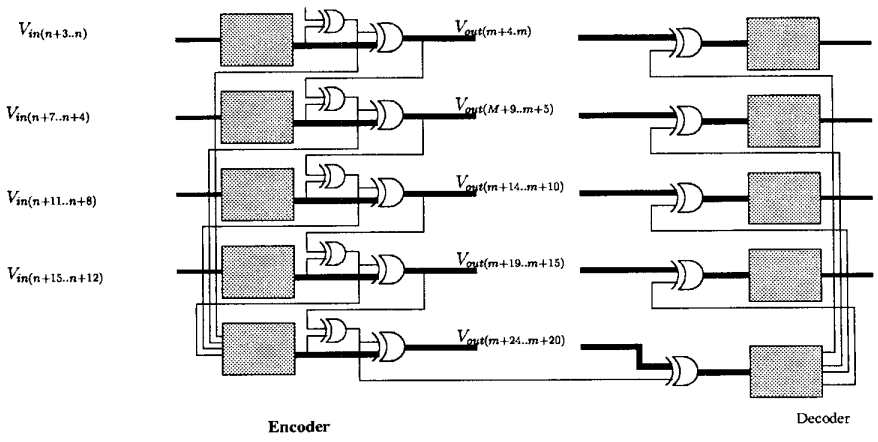

Figure 2. 16-bit CODEC structure

Based on the previous discussion, for a 4-bit bus, all $2^4 = 16$ vectors can be mapped to 16 5-bit vectors which are 2-C vectors. There are no forbidden patterns within any of the 5-bit vectors output by the encoder.

In practice, forbidden pattern can occur **across the group boundaries**. For instance, consider two encoded adjacent groups with data $\{11110\}\{11100\}$. Now there is a forbidden pattern between these groups (shown underlined). To prevent this from occurring, we invert the outputs of the second encoder if $v_{j,5} \neq v_{j+1,1}$ where j is the group index. The final output for our example will therefore be $\{11110\}\{00011\}$. An **group complement bit** is transmitted to enable the decoder to correctly decode the transmitted vector. The entire set of group complement bits are transmitted in a separate group, as shown in Figure 2.

In the above example, we need a total of 26 bits to be transmitted (for a 16-bit bus). This give an overhead of 62.5%.

3.3. Proof of Correctness

Definition 10 *S_{v5} is defined as the set of all possible valid 5-bit vectors.*

The column marked "output" in Table 1 is the set S_{v5}. Note that S_{v5} is closed under complementation.

In the encoder of Figure 2, consider two adjacent groups of encoded bus signals. Let these encoded inputs be $\{v_{j,1}, v_{j,2}, v_{j,3}, v_{j,4}, v_{j,5}\}$ and $\{v_{j+1,1}, v_{j+1,2}, v_{j+1,3}, v_{j+1,4}, v_{j+1,5}\}$.

- By construction, $\{v_{j,1}, v_{j,2}, v_{j,3}, v_{j,4}, v_{j,5}\} \in S_{v5}$ and $\{v_{j+1,1}, v_{j+1,2}, v_{j+1,3}, v_{j+1,4}, v_{j+1,5}\} \in S_{v5}$. Therefore, no forbidden transition occurs in either of these groups.
- Additionally, no forbidden transition occurs across group j and $j+1$ since the encoding algorithm complements $v_{j+1,i} (1 \leq i \leq 5)$ whenever $v_{j,5} \neq v_{j+1,1}$. Since S_{v5} is closed under complementation, no forbidden patterns result within the complemented group. Also, since the complementation ensures that $v_{j,5} = v_{j+1,1}$ in the transmitted data, no forbidden transitions can occur across groups j and $j+1$.

4. Eliminating 4-C Cross-talk Sequences

In this section, we discuss a scheme to eliminate $4 \cdot C$ cross-talk sequences, with a lower overhead than the previous scheme.

4.1. A $4 \cdot C$ Encoding Technique

In this method, we split a bus into **groups** of adjacent bus signals. The size of each group is 3. These group signals are referred to as **group data signals**. For each group, we add an additional signal called a **group complement signal**. When a group complement signal transitions, the bus receiver complements all group data signals corresponding to that group.

For vector V_k, we denote the group data signals of group i as $d^k_{i,j}$ (for $j = 1, 2, 3$) and the group complement signal for group i as c^k_i.

For two consecutive vectors V_k, V_{k+1}, let $d^k_{i,1} = v_1$, $d^k_{i,2} = v_2$, $d^k_{i,3} = v_3$ and $c^k_i = v_4$. Also, let $d^{k+1}_{i,1} = v_5$, $d^{k+1}_{i,2} = v_6$, $d^{k+1}_{i,3} = v_7$ and $c^{k+1}_i = v_8$. In that case, we denote the group transition as $(v_1, v_2, v_3, v_4)^k_i \rightarrow (v_5, v_6, v_7, v_8)^{k+1}_i$. If we refer only to the group data signals of group i, their transition is denoted as $(v_1, v_2, v_3)^k_i \rightarrow (v_5, v_6, v_7)^{k+1}_i$.

Whenever the signals v_1, v_2, v_3 satisfy the conditions of a forbidden transition and $v_1 = \overline{v_5}, v2 = \overline{v_6}, v_3 = \overline{v_7}$ the group data is transmitted as $(v_1, v_2, v_3, v_4)_i^k \rightarrow (v_1, v_2, v_3, \overline{v_4})_i^{k+1}$. The receiver complements these group data signals since the group complement signal performs a transition. Accordingly, correct data is recovered.

4.1.1. An Example

Consider a 9-bit bus. Assume that no encoding was performed, and consider two successive vectors on this bus as indicated below:

$(010\ 101\ 010)^k \rightarrow (101\ 010\ 101)^{k+1}$

Spaces are provided above to indicate group partitions. Now assume that all group complement signals are 0. Then the transmitted 12-bit vector sequence is:

$(0100\ 1010\ 0100)^k \rightarrow (0101\ 1011\ 0101)^{k+1}$.

Note that in the transmitted vector above, none of the group data signals perform a transition. In this fashion forbidden transitions never occur within a group. Again, spaces are provided to indicate group partitions.

On further reflection, we realize that forbidden transitions can occur *across* groups if this scheme is utilized. Consider a 6-bit bus. Assume that group complement signals are 0 to start with. Suppose the un-encoded bus transitions are:

$(010\ 101)^k \rightarrow (101\ 011)^{k+1}$

After encoding, the transmitted vector would be:

$(0100\ 1010)^k \rightarrow (0101\ 0110)^{k+1}$

Note that a forbidden transition occurs between $c_1^k, d_{2,1}^k$ and $d_{2,2}^k$ in the above case.

We observe that such a forbidden transition can *only* occur when the group complement signal transitions. More precisely, this kind of forbidden transition occurs when the group complement signal of group i performs a $0 \rightarrow 1$ (or $1 \rightarrow 0$) transition and the data signals of the group $i+1$ perform a $(10v)_{i+1}^k \rightarrow (01v)_{i+1}^{k+1}$ (or $(01v)_{i+1}^k \rightarrow (10v)_{i+1}^{k+1}$) transition, where $v \in \{0,1\}$. Therefore this condition can be avoided by modifying the original encoding scheme in the following manner.

Whenever $0(or1) = c_i^k \neq c_i^{k+1}$, and the group data signals of group $i+1$ are of the form $(10v)_{i+1}^k \rightarrow (01v)_{i+1}^{k+1}$ (or $(01v)_{i+1}^k \rightarrow (10v)_{i+1}^{k+1}$) where $v \in \{0,1\}$, we complement c_{i+1}^{k+1} and the group data signals of group $i+1$.

This complicates bus encoding, possibly forcing a rippling of data in the group complement logic and resulting in a slower encode.

input	output
0000	00000
0001	00001
0010	00110
0011	00011
0100	01100
0101	00111
0110	01110
0111	01111
1000	11111
1001	11110
1010	11001
1011	11100
1100	10011
1101	11000
1110	10001
1111	10000

Table 1. 4→5 Encoder Input-Output Mapping

4.2. Improved encoder

The above encoder exhibits a 33% overhead, and removes all $4 \cdot C$ transition. However, the ripple effect becomes significant when the bus becomes wider. Pipelining would need to be implemented to guarantee the overall data rate, thus increasing the latency.

To improve this, we present another encoder that is very similar to what has been discussed earlier. The new encoder still uses 33% overhead to get rid of all $4 \cdot C$ transitions. The difference is that there is no rippling between groups and all the encoding is done in a single stage.

The input bus is again split into 3-bit groups. The values of each group are compared with their previously transmitted values. The group data (or its complement) is transmitted based on the output of the comparator. Again, one group complement bit is transmitted per group.

The comparator implementation is slightly different. We check the first and the last two bits of the group. If a transition in opposite direction happens in either the first two bits or the last two bits (or both), the complementing condition is satisfied . For example, a '01x' $\rightarrow$ '10x' transition triggers the complementing, a 'x10' $\rightarrow$ 'x01' does so too.

This indeed removes all the $4 \cdot C$ transitions because whenever c_i has a transition, $d_{i,3}$ is guaranteed to be stable. Therefore, the capacitance between c_i and $d_{i,3} = 1 \cdot C$, and as a result, no $4 \cdot C$ sequence can be constructed across the two groups.

We can see that each group is totally independent of other groups and therefore all the encoding is done in a single step, thus improving circuit speed.

5. Experimental Results

Table 2 reports the worst-case delay among the bus signals. The results were generated using SPICE [4]. A 0.1μm process was assumed, and buses were assumed to be routed on Metal4. Wiring parasitics were obtained from [1] and wires were modeled as distributed RC transmission lines. In Table 2, 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.

Based on this table, we can see that the scheme of Section 3.2 would reduce the worst case delay by at least a factor of 2, regardless of bus trace length and driver strength. This results in significant savings for longer buses. In particular, for a 2cm bus driven by a 60× minimum driver, the worst case delay drops from 900 to 400 ps.

trc_length	bufsize	0c	1c	2c	3c	4c
10mm	30×	<100	200	350	550	750
10mm	60×	<100	100	250	400	500
10mm	120×	<100	120	170	300	350
20mm	30×	100	300	600	1000	1600
20mm	60×	100	250	400	650	900
20mm	120×	<100	150	300	550	750

Table 2. Delay Comparison (ps)

We implemented the encoder of Figure 2 and observed that the encoding delay was approximately 250 picoseconds. The decoder delay was determined to be approximately 250 picoseconds.

As a result, the overall delays on the bus are reduced for longer on-chip buses with reasonably sized drivers, even when encoding and decoding delay is accounted for. However, in case of heavily pipelined systems, the maximum data-rate is significantly improved by using our encoding schemes. In such pipelined systems, the encoding/decoding delays are unimportant.

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

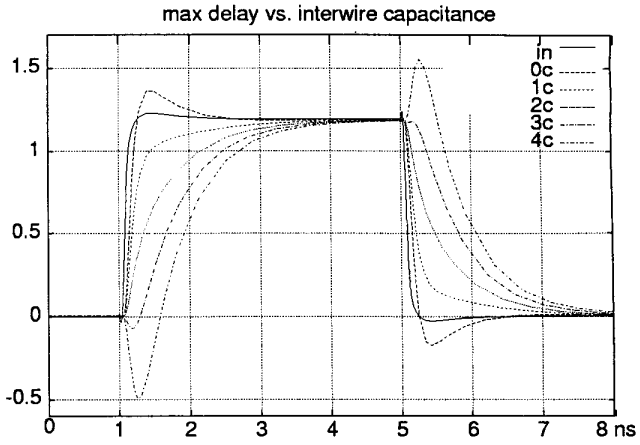

Figure 3. Sample SPICE Waveforms (60× driver, 2cm trace length, 0.1μ process)

6. Conclusions

Cross-talk between wires of an on-chip bus is becoming a significant problem in deep sub-micron IC design. Cross-talk can result in significant delay variations as well as signal integrity problems.

In this work, we classified cross-talk between wires in a bus, into several categories. Using this classification, we showed theoretical bounds on the number of additional bus bits required in order to eliminate $3 \cdot C$ and $4 \cdot C$ cross-talk sequences among bus wires. We presented a practical implementation of an encoding technique with a 62.5% bit overhead.

We presented another technique to eliminate $4 \cdot C$ cross-talk sequences. This technique exhibits a smaller overhead of 33%, with a less aggressive cross-talk reduction ability than the above method.

Experimental results were reported for these scheme, and we demonstrate at least 50% improvement in the worst-case delay in the bus wiring using the more aggressive techniques. With the technique that eliminates $4 \cdot C$ cross-talk, the delay improvements were less dramatic.

Further experiments are ongoing. In a highly pipelined system like a network processor, the encoding and decoding overheads are not significant. Initial experiments demonstrate that the encoding overheads are smaller than the gains offered by the cross-talk reduction schemes that we have proposed. Our gains are more significant for longer buses, making the technique attractive for large ICs.

7. Acknowledgements

The authors would like to thank Ericsson Wireless Communications and Jasmine Networks for their support of this research effort.

References

[1] S. P. Khatri, *Cross-talk Noise Immune VLSI Design using Regular Layout Fabrics*. PhD thesis, UC Berkeley, Dec 1999.

[2] K. Kim, K. Baek, N. Shanbhag, C. Liu, and S.-M. Kang, "Coupling-driven signal encoding scheme for low-power interface design," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, Nov 2000.

[3] P. Sotiriadis and A. Chandrakasan, "Bus energy minimizaiton by transition pattern coding (TPC) in deep sub-micron technologies," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, Nov 2000.

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