A Crosstalk Aware Interconnect with Variable Cycle Transmission *

Lin Li, N. Vijaykrishnan, Mahmut Kandemir, and Mary Jane Irwin
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802, USA
{*lili, vijay, kandemir, mji*}*@cse.psu.edu*

Abstract

Crosstalk between wires, caused by increased capacitive coupling, is considered one of the major factors that affect the performance of interconnects such as buses. The data-dependent nature of crosstalk-induced delays necessitates bus cycle time to be designed for the worst case crosstalk. However, this pessimism incurs a significant performance penalty. Consequently, we propose a crosstalk aware interconnect that uses a faster clock and dynamically controls the number of cycles required for transmission based on the estimated delay of the data pattern to be transmitted. In order to accomplish this, we designed a crosstalk analyzer circuit that is incorporated into the sender side of the bus and support a variable cycle transmission mechanism. We evaluate the effectiveness of the proposed scheme focusing on the on-chip buses of a microprocessor and by using the SPEC2000 benchmarks. The experimental results show that the proposed approach improves performance by 31.5% as compared to the original pessimistic approach. Furthermore, we employ a coding optimization to enhance the effectiveness of the proposed approach. We also show that the proposed scheme is an area-efficient approach to improving performance as compared to other crosstalk reduction schemes.

1. Introduction

With dramatic scaling in feature size, the propagation delay of long on-chip interconnect is becoming more significant than the delay of gates in deep sub-micron technology. Specifically, crosstalk between signals, caused by increased capacitive coupling, is considered one of the major problems that affect the timing of signals and consequently the cycle time must be increased. This, in turn, can lead to performance degradation and functional failures [8]. Therefore, alleviating the impact of crosstalk on the propagation delay of interconnect is very important in high performance system design.

There are many schemes employed at different levels to reduce the impact of crosstalk. Net ordering and buffer insertion are employed in the physical design of interconnects [9, 3]. Shielding of the wires and increasing the inter-wire spacing are other options explored for reducing the impact of crosstalk [4]. Another approach transforms the actual data to be transmitted to a coded form in order to reduce crosstalk [12]. Active shielding method employs two shielding wires on both sides of the target wire, and keeps the same transition direction as the target wire for fast propagation [7]. However, the above schemes (shielding, active shielding, increased spacing and the coding schemes) incur a significant area overhead. Furthermore, these methods determine the delay associated with data transmission based on the worst-case data transition pattern possible. However, the impact of crosstalk on the delay is data-dependent [10] and different combinations of transition directions of a specific wire and its adjacent wires induce different delays.

Operating assuming worst-case delay can be overly pessimistic for a significant portion of the data transmissions. This pessimism is undesirable because it increases cycle time, area and power [8]. One approach that has been recently proposed to eliminate the pessimistic safety margins due to factors such as (but not limited to) data-dependent latencies is [5]. Here, the supply voltage is tuned below the normal operating voltage to meet the average case delay in order to save energy. Possible errors are monitored. When delay induced by a particular data pattern exceeds that supported by this supply voltage, error correction is required. The essence of their approach is to trade-off the energy gain from voltage scaling and the energy overhead incurred from dynamic error correction. In contrast to this approach that adjusts the voltages reactively to correct errors when delays exceed that of the average case, we design a crosstalk aware

* This work was supported in part by grants from MARCO/DARPA GSRC-PAS, NSF Grants 0103583, 0082064 and CAREER Awards 0093082, 0093085.

1530-1591/04 $20.00 (c) 2004 IEEE

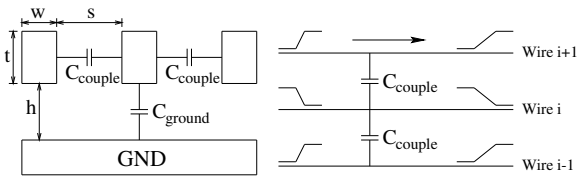

Figure 1. (a) Wire capacitances (b) Worst case propagation delay for wire i occurs when wire i transitions in the opposite direction of adjacent wires $i+1$ and $i-1$.

interconnect that uses a faster clock and dynamically controls the number of cycles required for transmission based on the estimated delay of the data pattern to be transmitted.

The main contributions of this work are as follows:

- We design a crosstalk aware interconnect by adding a crosstalk analyzer circuit to the sender side of the bus. The crosstalk analyzer determines the delay associated with the transition pattern comparing the previous data and current data to be transmitted. The output of the analyzer is used to dynamically control the number of cycles required for transmission in order to improve performance. We denote this approach as DYN. Experimental results show that this approach, using a faster clock with variable cycles for data transmission, provides significant performance gains as compared to using a single clock based on the worst-case crosstalk. We also compare the proposed scheme with other crosstalk reduction techniques and show that the proposed technique is an area-efficient mechanism to enhance performance.

- In order to further improve the performance of the proposed approach, we employ bus coding to transform many of the data transitions into those that have a lesser impact due to crosstalk, thereby enabling them to be transmitted in fewer cycles of the multi-cycle transmission. This coding scheme provides up to 10.5% performance improvement over DYN and 38.7% improvement over the original pessimistic approach.

The remainder of this paper is organized as follows. In the next section, the relationship between the different transition patterns and the delay of interconnect is explained. Section 3 presents the proposed crosstalk aware interconnect with variable cycle transmission method. The experimental framework and simulation results are provided in Section 4. In Section 5, we combine the DYN method with a bus coding method. Section 6 provides conclusions.

Table 1. Capacitance variations based on transition patterns.

Group	C_{total}	Transition Patterns			
1	0	$(-,-,-)$ $(-,-,\uparrow)$ $(\uparrow,-,\uparrow)$	$(-,-,\downarrow)$ $(\uparrow,-,\downarrow)$	$(\uparrow,-,-)$ $(\downarrow,-,\uparrow)$	$(\downarrow,-,-)$ $(\downarrow,-,\downarrow)$
2	C_{ground}	$(\uparrow,\uparrow,\uparrow)$	$(\downarrow,\downarrow,\downarrow)$		
3	$C_{ground}+C_{couple}$	$(-,\uparrow,\uparrow)$	$(-,\downarrow,\downarrow)$	$(\uparrow,\uparrow,-)$	$(\downarrow,\downarrow,-)$
4	$C_{ground}+2C_{couple}$	$(-,\uparrow,-)$ $(\uparrow,\uparrow,\downarrow)$	$(-,\downarrow,-)$ $(\uparrow,\downarrow,\downarrow)$	$(\downarrow,\uparrow,\uparrow)$	$(\downarrow,\downarrow,\uparrow)$
5	$C_{ground}+3C_{couple}$	$(-,\uparrow,\downarrow)$	$(-,\downarrow,\uparrow)$	$(\uparrow,\downarrow,-)$	$(\downarrow,\uparrow,-)$
6	$C_{ground}+4C_{couple}$	$(\uparrow,\downarrow,\uparrow)$	$(\downarrow,\uparrow,\downarrow)$		

2. Crosstalk Model

The various capacitances associated with wires are depicted in Figure 1(a). Here, C_{couple} is the capacitance between two adjacent wires and C_{ground} is the capacitance between a wire and the ground. The values of C_{couple} and C_{ground} are determined by the technology parameters such as wire width (w), spacing (s), wire thickness (t), height (h) and the length of the wires. As feature size scales down to deep sub-micron, the coupling capacitance contributes to a larger portion of the total capacitance.

Due to the capacitance and resistance on the wires, signal switching incurs a propagation delay. The propagation delay of the RC circuits can be calculated by

$$T_{delay} \propto R_{total} * C_{total} = R_{total} * (C_{ground} + n * C_{couple}) \quad (1)$$

In the above formula, n is dependent on the pattern of transition direction of the target wire and its two adjacent wires. We use the triplet (d_{i-1}, d_i, d_{i+1}) to represent the transition direction for adjacent wires $i-1$, i, and $i+1$. Here, the target wire for crosstalk analysis is the wire i. '$\uparrow$', '$\downarrow$', and '-' are used to represent transitions from 0 to 1, from 1 to 0, and no transition, respectively. The relation between C_{total} and transition patterns of these three wires are analyzed in [10] and are shown in Table 1[1]. Based on the total capacitance, we can get the delay of the target wire using the above formula.

For any given three wires, we divide all transition patterns into different groups as shown in Table 1. The patterns in the same group incur the same delay for transmission. The delay increases from minimum to maximum as we move from Group 1 to Group 6. The worst case delay is incurred by the transition pattern in Group 6, which happens when the direction of the transition of the target wire is opposite to those of its two adjacent wires (See Figure 1(b)). The worst case delay is determined by $(C_{ground} + 4C_{couple}) * R_{total}$. Since transition patterns in Group 5 and Group 6 are the most problematic ones, they are the focus of our approach. Based on the delay analysis of each wire, the delay

1 The capacitance for the boundary wires can be found in [10].

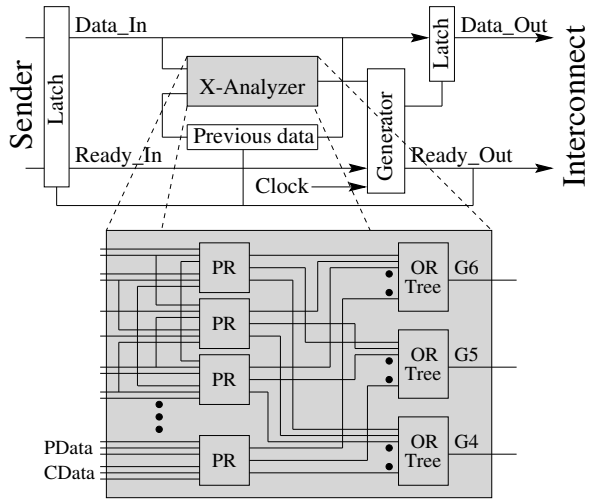

Figure 2. DYN method implementation.

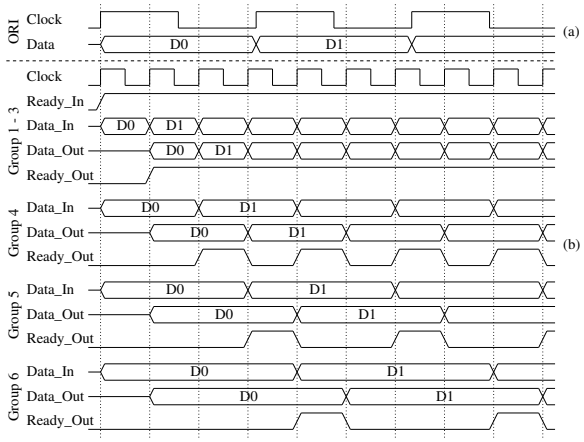

Figure 3. Timing sequence for (a) original pessimism method & (b) DYN method.

of an interconnect with 32 wires is computed as the maximum delay among these 32 wires.

3. Crosstalk Aware Interconnect

3.1. Crosstalk Analyzer

It is clear from Table 1 that the propagation delay of the wires is determined by the changes in value transmitted in the previous and current cycles. In our approach, using a register, the sender portion of the bus stores the previous data. When the new data arrives, the sender identifies the transition pattern between previous data and current data and determines the corresponding delay group (See Table 1) for each wire. The wire that belongs to the largest group determines the delay of the entire 32-bit bus. Therefore, the impact of crosstalk is captured by the sender before the transmission begins. This in turn provides the opportunity for the system to adjust for the different degrees of crosstalk impact.

To capture the transitions on the bus and identify the delay grouping, we incorporated an extra circuit, called X-analyzer, in the sender side of the bus as shown in the bottom part of Figure 2. Here, *PR* stands for the Pattern Recognition unit, which determine whether the 3-bit transition pattern belongs to Group 4, 5 or 6. Note that we limit the number of distinct groups identified to reduce the implementation complexity. The *OR Trees* are used to combine the signals indicating the patterns recognized for each of the 32 wires into signals $G4$, $G5$ and $G6$. These signals indicate whether patterns in Group 4, 5 or 6 exist in the whole 32-bit data. The signal $G4$, $G5$ and $G6$ is used for variable cycle transmission method as discussed in the next subsection.

3.2. Variable Cycle Transmission

The bus clock cycle is normally designed for the worst-case scenario of crosstalk determined by $(C_{ground} + 4C_{couple}) * R_{total}$. But there might be many transmissions which incur a delay that is less than that of the worst case. Therefore, using a single long clock cycle (See Figure 3(a)) may not be performance efficient. Hence, the proposed approach uses a variable cycle transmission whose basic idea is to use multiple short clock cycles instead of the original long clock cycle, where the number of clock cycles is determined by the X-analyzer based on the delay group (See Figure 3(b)).

The implementation of our method is shown in Figure 2. The sender of the original bus has 32 wires for data and 1 wire for the control signal *Ready_Out*. When the *Ready_Out* signal is high, the receiver of the bus knows that the data is ready. By controlling *Ready_Out* signal, the cycles used to transmit data can be changed dynamically. The timing sequences for the data transitions belonging to four different categories (for group 1-3, group 4, group 5 and group 6) are shown in Figure 3.

The clock cycle $T_{clk-ori}$ in the original method (denoted ORI) is at least $(C_{ground} + 4C_{couple}) * R_{total}$, independent of the group to which the transition belongs. In contrast, a short clock cycle of $T_{clk-dyn} = (C_{ground} + C_{couple}) * R_{total}$ is used in our approach. The X-analyzer incurs a one cycle latency to identify the transition pattern.

When data and *Ready_In* signal are issued from the sender, the X-analyzer determines the group to which the current transition belongs and correspondingly sets the outputs G4, G5 and G6. In the next cycle, the data (Dx) is placed on the bus and the signals G4, G5 and G6 determine when *Ready_Out* is placed on the bus. The delay between when the data and *Ready_Out* are placed on the bus allows

Table 2. Characteristics of different data transmission methods.

	C_{ground} (fF/mm)	C_{couple} (fF/mm)	Number of Wires	Normalized Cycle Time	Normalized Area 2mm	5mm	10mm	Extra Energy
ORI	36.3	115.1	32	3.28	100	100	100	-
CPC	36.3	115.1	53	2.76	174	170	168	12.6 mW
DYN	36.3	115.1	33	1.00	132	113	106	32.1 mW
DBS	53.1	60.4	32	1.95	149	149	149	-
SHD	36.3	115.1	63	1.76	198	198	198	-

the necessary number of cycles for the data on the bus to settle at the receiver side. The *Ready_Out* signal itself uses active shielding and reaches the receiver side in a single bus cycle immaterial of the data transmitted. For data patterns from Group 4, Group 5, and Group 6, the *Ready_Out* signal is placed one, two and three cycles after the data is placed on the bus. Otherwise, *Ready_Out* is placed on the bus in the same cycle as the data. Once the *Ready_Out* is placed on the bus, the next data to be transmitted is fed into the X-analyzer and the process repeats. It must be observed that the latency incurred by the X-analyzer is overlapped with the cycle that it takes for the *Ready_Out* signal to traverse the bus.

4. Experiments

We designed the crosstalk analyzer in VHDL and synthesized it using the Synopsys Design Compiler with a target 0.25um library. The parameters of 0.25um technology are from [13], and the Berkeley predictive technology model [1] is used to calculate the total capacitance of the interconnects. *w*, *s*, *t*, and *h* are set to 0.32um, 0.32um, 0.58um, and 0.70um, respectively. We also model three different wire lengths of 2mm, 5mm and 10mm to capture respectively short, medium and longer wires in the target technology. Short wires are representative of the longest connection within a module; medium wires can model on-chip buses such as the data bus and instruction bus from the datapath to L1 caches; and long wires are representative of on-chip multiprocessor buses. We use Simplescalar 3.0 [6] with default configuration parameters for the processor and the SPEC2000 CINT [2] benchmark suite to simulate the performance of the data bus connecting the processor datapath and L1 D-cache. While we have performed similar analysis for the instruction cache buses and observed similar benefits with our approach, we focus only on the data bus of L1 D-cache for brevity in the rest of this paper.

4.1. Methodology

In addition to the original transmission method (ORI) and our DYN method, we also implemented three other methods, CPC, DBS, and SHD, for comparison. These three methods are also used to reduce the impact of crosstalk and are shown in Figure 4 along with ORI.

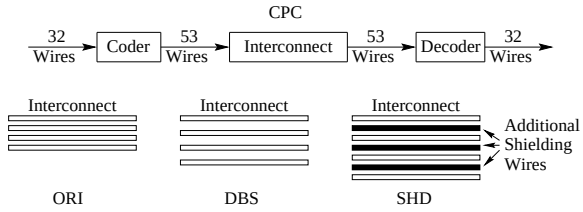

Figure 4. Crosstalk Prevention Coding, Double Spacing, and Shielding.

CPC stands for Crosstalk Prevention Coding [12]. This method codes the source data into transmission data via extra circuitry in the form of coder/decoder on the sender/receiver, respectively. There are more wires for transmitting coded data than that for source data. In the transmission data, there are no adjacent transitions in the opposite direction. Hence, all the transition patterns in Groups 5 and 6 are eliminated. For a 32-bit source data, 46 wires are required for data transmission based on the methodology proposed in [12]. For implementation purposes, we divide the wires into multiple smaller groups and build the coder and decoder for these smaller groups. The different groups of the wires are separated by shielding wires. Specifically, we coded every 3 bits into 4 bits. Hence, a total of 53 wires are required for transmitting the original 32-bit data.

DBS stands for the Double Spacing method [4]. Increasing the space between adjacent wires can reduce the coupling capacitance and the total capacitance. Although the worst case of transition patterns still exist, the clock cycle is reduced due to the smaller total capacitance. In our experiments, the space between the adjacent wires were doubled, which means parameter *s* is changed to 0.64um.

SHD stands for the Shielding method [4]. SHD inserts one V_{dd} or *Ground* wire between every two wires. Since there is no transition on the shielding wires, transition patterns in Groups 5 and 6 are eliminated. Therefore, the worst case delay reduces to $(C_{ground} + 2C_{couple}) * R_{total}$.

4.2. Experimental Results

The main characteristics of the different data transmission methods are shown in Table 2. Coupling capacitance

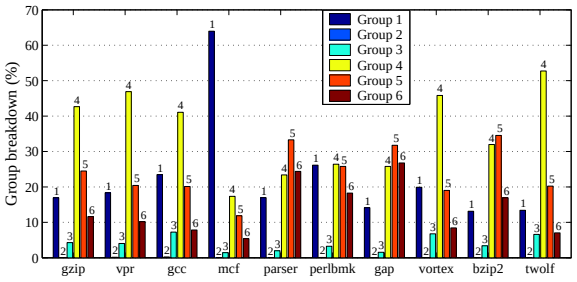

Figure 5. Distribution of transition patterns for Group 1 to 6.

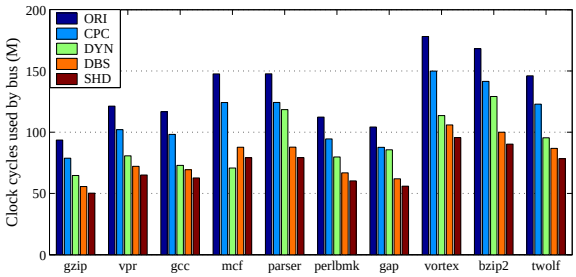

Figure 6. Performance of different data transmission methods.

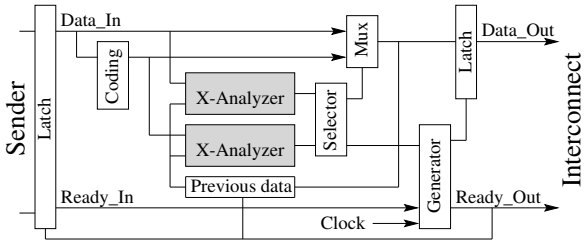

Figure 7. DYN + Bus-invert coding technique.

and ground capacitance are listed in the second and third columns, assuming minimum spaced wires. Since DBS has different space parameters than other methods, its capacitances are different. We can observe that coupling capacitance contributes to a large portion of the total capacitance. The number of the wires required for each method is listed in the fourth column. The cycle times for the different transmission methods are normalized to that of DYN and listed in the fifth column. CPC needs extra circuits as coder and decoder, and DYN needs extra circuits as crosstalk analyzer. We can observe that CPC, DBS, and SHD incur a large area overhead (70%, 49%, and 98% for 5mm bus respectively) as compared to ORI. In comparison, DYN only increases area by 13%.

Energy overheads for methods of CPC and DYN are also listed in the last column. As the crosstalk analyzer circuit in DYN remains the same when interconnect length (and consequently interconnect area) increases, the relative area overhead incurred by DYN decreases with the length of the interconnects. On the other hand, the schemes that use more wires or wire spacing, the relative area overhead remains the same with the change in length. This means that the area overhead of the DYN method would decrease for the longer interconnection. As shown in Table 2, the area overhead of DYN reduces from 32% to 6% when moving from a 2mm to a 10mm interconnect.

Figure 5 shows the distributions of different groups of transition pattern for each benchmark. We see that in most cases, the worst case (Group 6) is not the dominant case. We also see that Group 4 is a large contributor. The distributions are determined by the intrinsic characteristics of the benchmarks. The average distributions are 22.64%, 0.05%, 4.06%, 35.4%, 24.2%, and 13.7% for Group 1 through Group 6.

Figure 6 gives the results of bus performance for different transmission schemes normalized in terms of number of bus cycles. Note that the impact of different clock cycles is captured by the normalization. From left to right are the methods ORI, CPC, DYN, DBS, and SHD. Since SHD places a V_{dd} wire between every two adjacent wires, the impact of crosstalk on the delay is reduced to the minimum degree. Therefore, the performance of SHD is the best among all the methods we experimented. But it incurs the most area overhead, which is almost twice the area of the original method. DBS (due to spacing) and CPC (due to additional wires) also come with area penalties. Our dynamic approach provides an average of 31.5% performance improvement over the original scheme. In comparison, the average performance improvements due to CPC, DBS, and SHD methods are 15.9%, 40.5%, 46.3%, respectively.

Among the benchmarks, the DYN is better than all other methods for *mcf* and near CPC for *gap*. From Figure 5, we find that the reason is that Group 5 transition patterns are dominant in *gap*, but Group 1 are dominant in *mcf*. This means that the performance improvement depends on the distribution of the transition patterns.

5. Influence of Bus Coding

Based on the experimental results in the last section, the improvement of interconnect performance for the DYN method depends on the distribution of transition patterns. The more the transitions in Groups 1 to 3, the larger the improvement in performance that can be achieved. In our DYN method, the transmitted data is the same as the source data. Therefore, the distribution of the transition pattern is determined by the intrinsic characteristics of the benchmarks.

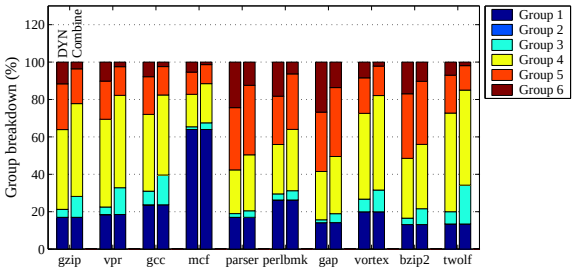

Figure 8. Distribution of transition patterns of DYN and DYN + Bus-invert coding. In each bar, Groups 1 to 6 are shown from bottom to top.

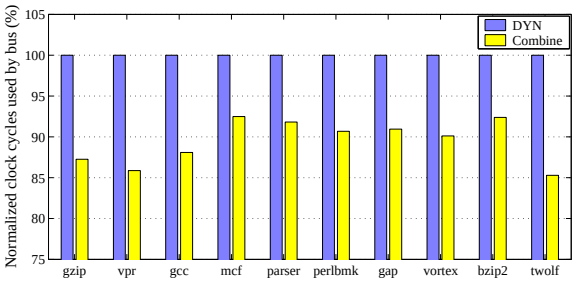

Figure 9. Performance of DYN + Bus-invert coding.

In order to transform this distribution to increase the transitions that belong to the lower groups (1-3), we employed the bus-invert coding scheme [11] with our DYN method (See in Figure 7). Two crosstalk analyzers are used for determining whether the original data or the inverted data will result in smaller crosstalk delay. Accordingly, either the original or inverted data is selected for transmission.

Figure 8 shows the distributions of transition patterns when combining bus-invert coding with DYN method. We can observe that using bus-invert coding increases the transmissions in Groups 1, 3, and 4 by 0.06%, 5.9%, and 3.8%, and reduces transmissions in Groups 5 and 6 by 1.6% and 8.0%, respectively. In Figure 9, the consequent average performance improvement over DYN is 10.5% when using bus-invert coding. However, the additional circuitry incurs an area overhead of 22%, 12%, and 8% for 2mm, 5mm, and 10mm bus, respectively, as compared to DYN. Therefore, this coding approach is likely to be more attractive for medium to long wires.

6. Conclusion

Crosstalk induced delays are transition dependent. Designing bus cycle times based on worst-case crosstalk is overly pessimistic. In this paper, we design a crosstalk aware interconnect that uses a faster clock and dynamically controls the number of cycles required for transmission based on the estimated delay of the data pattern to be transmitted. The experimental results show that the proposed approach improves performance by 31.5% as compared to the original pessimistic approach (that always assumes the worst case). Furthermore, we employ a coding optimization to enhance the effectiveness of the proposed approach.

References

[1] Berkeley predictive technology model. http://www-device.eecs.berkeley.edu/˜ptm/interconnect.html.

[2] SPEC CPU2000 Benchmark. http://www.spec.org/.

[3] C. J. Alpert, A. Devgan, and S. T. Quay. Buffer insertion for noise and delay optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(11):1633–1645, 1999.

[4] R. Arunachalam, E. Acar, and S. R. Nassif. Optimal shielding/spacing metrics for low power design. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 167–172, 2003.

[5] D. Blaauw, T. Austin, and T. Mudge. Razor: Exploiting runtime error correction in microprocessor design. In *GSRC June 2003 Quarterly Workshop Presentations*, June, 2003.

[6] D. C. Burger and T. M. Austin. The SimpleScalar tool-set, Version 2.0. Technical Report 1342, Dept. of Computer Science, UW, June 1997.

[7] H. Kaul, D. Sylvester, and D. Blaauw. Active shields: a new approach to shielding global wires. In *Proceedings of the 12th ACM Great Lakes Symposium on VLSI*, pages 112–117, 2002.

[8] B. Kiani. Static crosstalk analysis assures silicon success. *EE Times*, June 5, 2002.

[9] J. D. Z. Ma and L. He. Formulae and applications of interconnect estimation considering shield insertion and net ordering. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 327–332, 2001.

[10] P. Sotiriadis and A. Chandrakasan. Reducing bus delay in sub-micron technology using coding. In *IEEE Asia and South Pacific Design Automation Conf.*, pages 109–114, 2001.

[11] M. R. Stan and W. P. Burleson. Bus-invert coding for low-power i/o. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(1):49–58, 1995.

[12] B. Victor and K. Keutzer. Bus encoding to prevent crosstalk delay. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 57–63, 2001.

[13] K. Yamashita and S. Odanaka. Impact of crosstalk on delay time and a hierarchy of interconnects. In *IEDM '98 Technical Digest*, pages 291–294, 1998.