A BUS ENCODING TO REDUCE CROSSTALK NOISE EFFECT IN SYSTEM ON CHIP

J.Venkateswara Rao[1] and A.V.N.Tilak[2]

[1]Department of Electronics and Communication Engineering, Vignan Institute of Technology & Science, Deshmukhi, Nalgonda Dist. A.P, India.
jvrao_07@yahoo.com
[21]Department of Electronics and Communication Engineering, Gudlavalleru Engineering College Gudlavalleru, Krishna Dist. A.P, India.
avntilak@yahoo.com

ABSTRACT

This paper proposes a new bus coding scheme for reducing the crosstalk in System on chip(soc). As circuit geometries become smaller, wire interconnections become closer together and taller, thus increasing the cross-coupling capacitance between nets. At the same time, parasitic capacitance to the substrate becomes less as interconnections become narrower, and cell delays are reduced as transistors become smaller. With circuit geometries at 0.25 micron and above, substrate capacitance is usually the dominant effect. However, with geometries at 0.18 micron and below, the coupling capacitance between nets becomes significant, making crosstalk analysis increasingly important for accurate timing analysis. We show experimentally that the proposed codes allow reducing crosstalk delay by at least 14% based on available data.

KEYWORDS

Crosstalk, Encoding, SOC, parasitic, coupling Capacitance, micron, Forbidden Pattern free

1. INTRODUCTION

As device geometries shrink, chip sizes increase, and clock speeds get faster, interconnect delay is becoming increasingly significant. Signal integrity is the ability of an electrical signal to carry information reliably and resist the effects of high-frequency electromagnetic interference from nearby signals. Crosstalk is the undesirable electrical interaction between two or more physically adjacent nets due to capacitive cross-coupling. As integrated circuit technologies advance toward smaller geometries, crosstalk effects become increasingly important compared to cell delays and net delays.

In particular, the propagation delay through long cross-chip buses is already proving to be a limiting factor in the speed of some designs, and this trend will only get worse. It has been shown that the delay through a long bus is strongly a function of the coupling capacitance between the wires. Especially detrimental to the delay is the Miller-like effect when adjacent wires simultaneously transition in opposite directions. When the cross-coupling capacitance is comparable to or exceeds the loading capacitance on the wires, the delay of such a transition may be twice or more that of a wire transitioning next to a steady signal. We call this delay penalty the "crosstalk delay". In some high-speed designs where crosstalk delay would have limited the clock speed, the technique of shielding was used. This involves putting a grounded wire between every signal wire on the bus. Although this certainly is effective in preventing crosstalk within the bus, it has the effect of doubling the wiring area. Cross-chip buses often must be routed in higher metal layers, which are scaled more slowly than the rest of the geometry in order to prevent an unacceptable increase in resistance. Thus, routing resources are scarce at these levels, and it can be difficult to justify doubling the bus width.

However, if we abstract the concept of shielding and just look at the signals on the wires of a shielded bus, we *think* of it a**s** a very simple bus encoding. Two wires are used for every data bit. A data bit of *"0"* is encoded as a *"00"* signal on the wires, and a **"**1**"** is encoded as "l0". The purpose of this "encoding" is to prevent adjacent wires from transitioning in opposite directions, and this particular encoding achieves that goal by forcing every other wire to a steady value. But the question arises: Are there other possible encodings that can achieve the same goal, but with fewer wires? Such encodings may require extra logic or memory elements, but as the speed of logic goes up and the relative area consumed by logic goes down, such a tradeoff seems increasingly valid. Indeed, such encodings exist. We will refer to them as "self-shielding" or "crosstalk-immune" codes. In this paper, we designed the codes based on the codes theoretical suggested in [7]. We will develop the theory behind crosstalk-immune coding, describe the fundamental capabilities and limitations that the theory implies, and give methods for generating optimal sets of code words.

The most aggressive of our encoding techniques actually speeds up a bus by exploiting crosstalk. In this encoding approach, if a bus signal rises (falls), then our encoding forces one of its neighbors to rise (fall) as well, while the other neighbor is static. As a result, we actually improve the rise time of the wire by utilizing crosstalk to our best. This is not possible with current approaches to alleviate the cross-talk problem in buses (which stagger bus signals in space and/or time to mitigate the cross-talk problem among bus signals). In recent times, with wiring delays increasing compared to gate delays [8], it is often the case that the critical delay in a circuit is determined by long buses. In such a case, buses could be encoded with the techniques described in this paper, allowing the design to be operated at a much greater frequency. A designer would gladly tolerate the bus size overhead involved with the use of our approach, in such a scenario.

Different approaches have been proposed for crosstalk reduction in the context of bus interconnects. Some schemes focus on reducing the energy consumption, some focus on minimizing the delay and other schemes address both. The simplest approach to address the inter-wire crosstalk problem is to shield each signal using grounded conductors. Khatri et al. in [12] proposed a layout fabric that alternatively inserts one ground wire and one power wire between every signal wire, i.e., the wires are laid out as . . . *VSGSVSGSVS* . . . , where *S* denotes a signal wire, *G* denotes a ground wire and *V* denotes a power wire. Compared to passive shielding, active shielding is a more aggressive technique that reduces the bus delay by up to 75% [13]. Ghoneima and Ismail in [14] investigated the theoretical optimum position for repeater insertions to achieve minimum delay. Again, this technique is a physical design technique and requires careful trace layout and repeater placement. The average delay can also be reduced using statistical approaches such as data packing and data permutation proposed by Satyanarayana et al. in [15]. All these techniques exploit the temporal coherency of bus data to minimize the average delay for data transmission.

This paper is organized as follows. Section 2 provides crosstalk delay effects. In Section 3, we discuss previously published approaches for solving this problem. In Section 4, we describe our approach of creating memoryless-based crosstalk canceling CODECs. In Section 5 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 size overhead incurred. In section 6, we compare our bus size overheads with those reported in [7], in which memoryless CODECs to eliminate crosstalk patterns were described. We conclude the paper in Section 7.

2. PRELIMINARIES OF CROSSTALK EFFECTS

Crosstalk can affect signal delays by changing the times at which signal transitions occur. For example, consider the signal waveforms on the cross-coupled nets A, B, and C in fig. 1. Because of capacitive cross-coupling, the transitions on net A and net C can affect the time at which the transition occurs on net B. A rising-edge transition on net A at the time shown in Fig. 1 can cause the transition to occur later on net B, possibly contributing to a setup violation for a path containing B. Similarly, a falling-edge transition on net C can cause the transition to occur earlier on net B, possibly contributing to a hold violation for a path containing B.

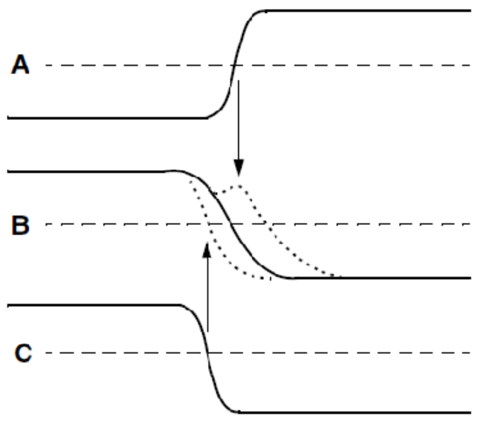

Figure 1. Transition Slowdown or Speedup Caused by Crosstalk.

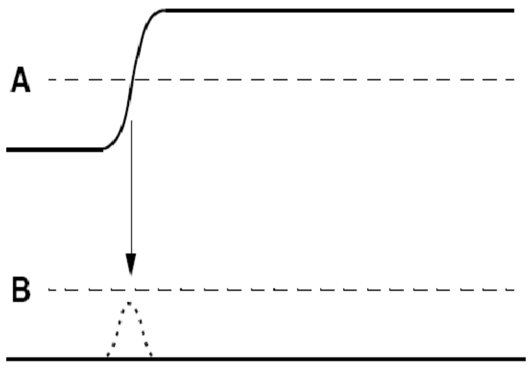

Figure 2. Glitch Due to Crosstalk.

The logic effects of crosstalk on steady-state nets are due to the cross-coupled nets as shown in fig. 2. Net B should be constant at logic zero, but the rising edge on net A causes a noise bump or glitch on net B. If the bump is sufficiently large and wide, it can cause an incorrect logic value to be propagated to the next gate in the path containing net B.

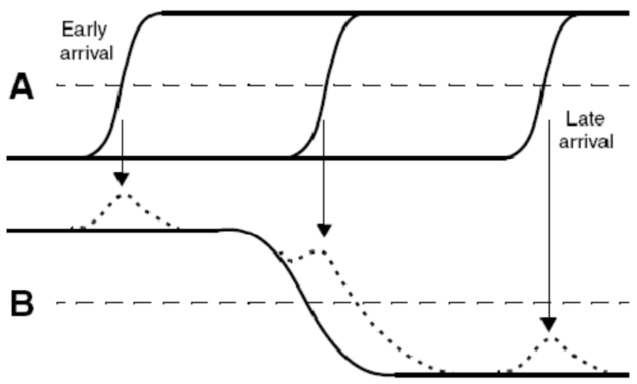

Figure 3. Effects of Crosstalk at Different Arrival Times.

A net that receives undesirable cross-coupling effects from a nearby net is called a victim net. A net that causes these effects in a victim net is called an aggressor net. Note that an aggressor net can itself be a victim net; and a victim net can also be an aggressor net. The terms aggressor and victim refer to the relationship between two nets being analyzed. The timing impact of an aggressor net on a victim net depends on several factors:

• The amount of cross-coupled capacitance

• The relative times and slew rates of the signal transitions

• The switching directions (rising, falling)

• The combination of effects from multiple aggressor nets on a single victim net

Figure 3 illustrates the importance of timing considerations for calculating crosstalk effects. The aggressor signal A has a range of possible arrival times, from early to late. If the transition on A occurs at about the same time as the transition on B, it could cause the transition on B to occur later as shown in the figure, possibly contributing to a setup violation, or it could cause the transition to occur earlier, possibly contributing to a hold violation. If the transition on A occurs at an early time, it induces an upward bump or glitch on net B before the transition on B, which has no effect on the timing of signal B. However, a sufficiently large bump can cause unintended current flow by forward-biasing a pass transistor. Similarly, if the transition on A occurs at a late time, it induces a bump on B after the transition on B, also with no effect on the timing of signal B. However, a sufficiently large bump can cause a change in the logic value of the net, which can be propagated down the timing path. In [7], they suggested the codes and encoder and decoder theoretically. We approached in different way with different Codes words and implemented using the Synopsys Design Compiler and Design Analyzer and provided the timing results.

3. Proposed Codes for Address Buses

Table 1. Codes for Address Buses.

Brute-Keutzer codes		**Our codes**	
000	0111	000	0000
001	0001	001	0010
010	1111	010	1010
011	0000	011	1110
100	0101	100	1000
101	0100	101	0011
110	1101	110	1011
111	1100	111	1111

The above Table 1 shows the codes proposed by Brute-Keutzer theoretically and codes proposed by us. All the three bit codes are encoded to four bit code words which eliminates the crosstalk. If there are 24 lines, then we have to incorporate an extra 8 lines to eliminate the crosstalk. All the codes are designed and simulated, synthesized and verified using Synopsys Tools.

4. Gate level Encoder and Decoder

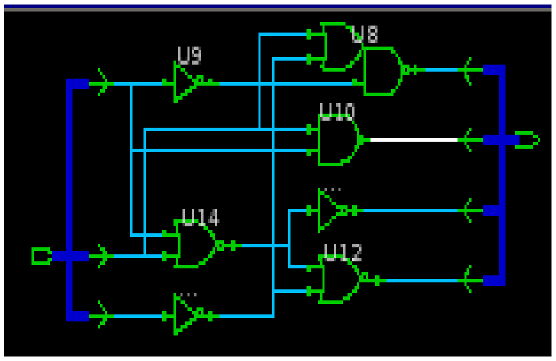

Figure 4. 3x4 encoder gate level schematic.

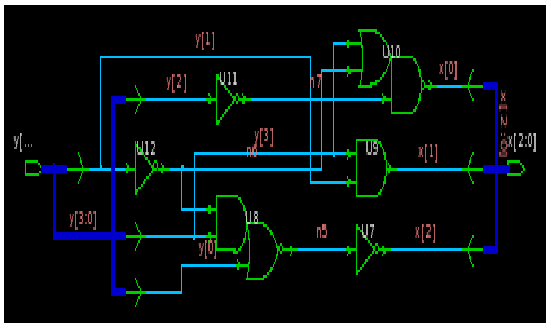

Figure 5. 4x3 decoder gate level schematic.

Figure 4 and 5 shows the gate level diagram of an encoder and decoder that could be implementing on System on chip buses. It is easy to recognize the encoder as a simple combinational circuit which occupies very less area and the decoder is a function only of the current code word with no feedback at all. This encoders and decoders can eliminate crosstalk due to coupling capacitances.

5. Experimental Results

Table 2. Timing Reports for Encoder.

Point	**Incr.**	**Path**
input external delay	0.00	0.00 f
x[1] (in)	0.00	0.00 f
U14/Z (NR2)	1.29	1.29 r
U12/Z (NR2)	0.28	1.57 f
y[0] (out)	0.00	1.57 f
data arrival time		1.57

The above timing report shows the data arrival time for the encoder from input side to output side. Prime Time reports the worst delay path from input to output. The x [1] is the input port of the encoder and the name in the bracket is the reference name for that port. Incr is the incremental path delay. The delay from input port, x[1] to the output of nor gate, (U14/Z) is 1.29 ns. The delay from the output of nor gate, (U14/Z) to the output of next nor gate, (U12/Z) is 0.28 ns. The output of nor gate, (U12/Z) is the output port of encoder, y(0). So, the total delay from input port, x[1] to output port y[0] is 1.5 ns. The f in the third column indicates a transition from 0→1 and r indicates a 1→0 transition.

Table 3. Timing Reports for Decoder.

Point	**Incr.**	**Path**
input external delay	0.00	0.00 r
y[1] (in)	0.00	0.00 r
U12/Z (IV)	0.32	0.32f
U8/Z (AO6)	1.22	1.54 r
U7/Z (IV)	0.18	1.72f
x[2] (out)	0.00	1.72 f
data arrival time		1.72

The above timing report shows the data arrival time for the decoder from input side to output side. Prime Time reports the worst delay path from input to output. y[1] is the input port of the decoder and the name in the bracket is the reference name for that port. Incr is the incremental path delay. The delay from input port, y[1] to the output of not gate, (U12/Z) is 0.32 ns. The delay from the output of not gate, (U12/Z) to the output of next and-or-invert gate, (U8/Z) is 1.22 ns. Next, the delay from the output of and-or-invert gate, (U8/Z) to the output of next not gate, (U7/Z) is 0.18 ns. The output of not gate, (U7/Z) is the output port of decoder, x(2). So, the total delay from input port, y[1] to output port x[2] is 1.72 ns. The f in the third column indicates a transition from 0→1 and r indicates a 1→0 transition.

6. Comparison to Other Techniques

This paper is based on the Theoretical concepts proposed in [7]. The encoder presented in fig 4 that takes the code words of our codes has a data arrival time of 1.5ns compared to encoder proposed in [7] has a data arrival time of 1.75ns. When the data arrival time is small, that creates a more positive slack. In the literature, there are a number of other techniques designed for combating crosstalk. Many of them, such as those described in [1], [2] and *[3],* employ creative routing strategies in order to minimize crosstalk delay within a data path or logic block. Our technique on the other hand, is intended for use with long, straight buses, and thus these routing schemes are not applicable to our domain of interest. [8] and [9] mention some techniques that are more relevant, such as skewing the timing of signals on adjacent wires, interleaving mutually exclusive buses, and precharging the bus. However, skewing requires careful, technology-dependent circuit design and brings up tricky timing issues, whereas our technique is technology-independent and fully synchronous, with the crosstalk immunity "correct by construction." Interleaving is a useful technique, but it cannot be used with buses that are allowed to transition on any and every clock cycle. Precharging a long bus can incur detrimental power costs, and is usually not an option. Probably the most common technique is simply using large repeaters to drive the Miller capacitance through brute force [10]. A quantitative comparison between our technique and optimally-sized repeaters is technology- and implementation-dependent. However, conceptually, using large repeaters is a power-hungry technique, and shielding is an area-hungry technique. Crosstalk-immune bus encoding avoids crosstalk delay with a modest impact on either area or power.

7. Conclusions

Cross-talk between wires of an on-chip bus is becoming a significant problem in deep sub-micron IC design. Crosstalk can result in significant delay variations as well as signal integrity problems. In this paper, we have introduced the concept of using data encoding to mitigate crosstalk delay on buses, and we presented a practical framework for understanding crosstalk immune coding. Current research is to design an efficient encoder and decoder to reduce crosstalk in on chip buses. The encoder and decoder proposed in our paper reduce crosstalk delay about 14% to that of available techniques.

Acknowledgements

The authors would like to thank Synopsys Tools providers. Without their support, research would have been a lot more painful experience than it already is.

References

[1] Hui Liu, Hong Shi, John Xie "Reduce Simultaneous Switching Jitter in Number, Spatial, Time, and Frequency Dimensions" IEEE Conf. on Electronic Components and Technology, page no**.** 1468–1473, 2010.

[2] Chaeho Chung, Soobum Lee, Byung Man Kwak, Gawon Kim, and Joungho Kim "Delay Line Circuit Design for Crosstalk Minimization Using Genetic Algorithm" IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, VOL. 27, NO. 3 March 2008.

[3] Chunjie Duan; Gulat. K.; Khatri S.P.; “Memory-based crosstalk canceling CODECs for on-chip buses” IEEE Int. Symp. on circuits and systems. pp. 1119-1122, 2006.

[4] P.Heydari, “Capacitance Coupling Noise in High Speed VLSI Circuits,” IEEE Trans. Computer aided design of integrated circuits, vol.24, march. 2005, page no**.** 478-489.

[5] P. Subrahmanya and R. Manimegalai and Muthyam, Madhu and V, Kamakoti “A Bus Encoding Technique for Power and Crosstalk minimization” IEEE International Conf. on VLSI Design, page no**.** 443 – 448, 2004

[6] Xiaoliang Bai and Sujit Dey “High level Crosstalk defect simulation methodology for System on Chip Interconnects” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, VOL. 23, NO. 9 September 2004.

[7] Bret Victor and and Kurt Keutzer “Bus Encoding to Prevent Crosstalk Delay” IEEE/ACM Int. Conf. page no. 57-63, 2001.

[8] T. Xue, E. Kuh, and D. Wang, Tost Global Routing Crosstalk Synthesis,” IEEE Trans. Computer-Aided Design, vol. 16, no. 12.

[9] J. Yim and C. Kyung, “Reducing Cross-Coupling among Interconnect Wires in Deep-Submicron Datapath Design,” Proceedings. 1999 Design Automation Conf., pp. 485-90. 1999

[10] K. Hirose and H. Yasuura, “A Bus Delay Reduction Technique Considering Crosstalk,” Proc. Design, Automation and Test in Europe Conf. and Exhibition 2000, pp. 441-5.2000.

[11] D. Li, A. Pua, P. Srivastava, and U. KO, “A Repeater Optimization Methodology for Deep Sub-Micron, High-Performance Processors,” Proc. Int. Conf. on Computer Design, VU1 in Computers and Processors. pp. 726-31. 1997

[12] S.P. Khatri, A. Mehrotra, R.K. Brayton, Ralf H.J.M. Otten, and A.L. Sangiovanni-Vincentelli. A novel vlsi layout fabric for deep sub-micron applications. In *Proceedings of Design Automation Conference*, pages 491–496. IEEE, 1999.

[13] C. Duan and S.P. Khatri. Exploiting crosstalk to speed up on-chip buses. In *Proceedings of Design Automation, and Test in Europe (DATE) Conference*, 2004.

[14] M. Ghoneima and Y. Ismail. Optimum positioning of interleaved repeaters in bidirectional buses. *Transactions on Computer-Aided Design*, 24(3):461–469, 2005.

[15] CN. Satyanarayana, M. Mutyam, and A.V. Babu. Exploiting On-Chip Data Behavior for Delay Minimization. In *Proceedings of International Workshop on System-Level Interconnect Prediction*, Austin, Texas, USA, 2007. IEEE/ACM.

Authors

J.Venkateswara Rao received B.Tech Degree from, VRSEC, Nagarjuna University, Guntur, in 2000. M.Tech. Degree from NITW, Warangal, India in 2003, pursuing Ph.D in the Department of Electronics and Communication Engineering, JNT University, Hyderabad, INDIA. Currently he is working as Associate Professor in the Department of Electronics and communication Engineering, VITS, Hyderabad, INDIA, His research interest includes on and off chip crosstalk noise reduction in VLSI Circuits.

Dr. Alapati V.Naga Ratna Tilak received his Ph.D. (Micro Electronics) from IIT Madras. Currently he is working as Prof. in ECE & Dean, Academic Affairs in Gudlavalleru Engineering College, Gudlavalleru, Krishna Dist. Andhra Pradesh. India. Currently he is guiding 4 Ph.Ds under JNT University. He is life Member of IEEE.