Bus Stuttering : An Encoding Technique to Reduce Inductive Noise in Off-Chip Data Transmission

Brock J. LaMeres
Design Validation Division
Agilent Technologies Inc.
Colorado Springs, CO 80907 USA
brock_lameres@agilent.com

Sunil P. Khatri
Electrical Engineering Department
Texas A&M University
College Station, TX 77843
sunil@ee.tamu.edu

Abstract

Simultaneous switching noise due to inductance in VLSI packaging is a significant limitation to system performance. The inductive parasitics within IC packaging causes bounce on the power supply pins in addition to glitches and rise-time degradation on the signal pins. These factors bound the maximum performance of off-chip busses, which limits overall system performance. Until recently, the parasitic inductance problem was addressed by aggressive package design which attempts to decrease the total inductance in the package interconnect. In this work we present an encoding technique for off-chip data transmission to limit bounce on the supplies and reduce inductive signal coupling. This is accomplished by inserting intermediate (henceforth called "stutter") states in the data transmission to bound the maximum number of signals that switch simultaneously, thereby limiting the overall inductive noise. Bus stuttering is cheaper than expensive package design since it increases the bus performance without changing the package. We demonstrate that bus stuttering can bound the maximum amount of inductive noise, which results in increased bus performance even after accounting for the encoding overhead. Our results show that the performance of an encoded bus can be increased up to 225% over using un-encoded data. In addition, synthesis results of the encoder in a TSMC 0.13μm process show that the encoder size and delay are negligible in a modern VLSI design.

1. Introduction

Advances in VLSI fabrication technologies have led to a dramatic increase in the on-chip performance of integrated circuits. The increase in IC performance is predicted by the International Technology Roadmap for Semiconductors (ITRS) [1] to continue doubling every 18 months, following Moore's Law, for at least the next several years [2]. However, package performance is predicted by the ITRS to only double over the next decade. This imbalance in performance expectations between the IC and the package is a major concern for system designers. The main limitation of the package performance is the parasitic inductance present in the level 1 (from IC die to package) and level 2 (from package to board) interconnects [3, 4, 5]. The inductance factors that affect signal speed and integrity are as follows:

- Supply bounce: Typically supply (V_{SS} and V_{DD}) pins are interspersed at regular intervals between signal pins. Every n^{th} pin is a V_{SS} or V_{DD}. The supply bounce is proportional to the number of pins switching low or high that return current through a particular V_{SS} and V_{DD} pin. In the case of Ground bounce, the noise is expressed as:

$$V_{bnc} = L \sum_{k} \left(\frac{di}{dt}\right) \quad (1)$$

  Where L is the self-inductance of the V_{SS} pin, and $\sum_k (\frac{di}{dt})$ is evaluated over the number of signal pins switching low that return current through the V_{SS} pin being evaluated. Since the placement of power and signal pins is regular, we can compute this quantity as half the number of signal pins switching low to the immediate right of the V_{SS} pin and half the number of signal pins switching low to the immediate left of the V_{SS} pin. Since each signal always has a V_{SS} pin to the left and to the right, we assume that if it switches low, then half the switching current is supplied by the V_{SS} pin to its left, and the other half by the V_{SS} pin to its right. In a similar manner, a supply voltage droop is encountered on V_{DD} pins as well.

- Glitching. If a signal pin j is static, then a glitch may be induced in its voltage due to neighboring pins which switch. This is governed by the expression

$$V^{j}_{glitch} = \sum_{k} \pm (M_{jk} \frac{di_k}{dt}) \quad (2)$$

  where i_k is the current in the k^{th} pin, and M_{jk} is the mutual inductance between the j^{th} pin being considered and the k^{th} pin. The sign of the coupled voltage is positive or negative depending on whether the k^{th} neighboring pin undergoes a rising or falling transition.

- Switching speed. When a signal is switching, coupled voltage due to neighboring signal pins that are also switching (Equation 2) can speed up, slow down, or not affect the switching speed of a particular signal. Since the sign of the coupled voltage from any neighboring signal depends on the polarity of its edge, equal and opposite magnitudes of coupled voltages from different neighbors can cancel and leave a net effect of zero on the victim signal. This behavior can be exploited such that only patterns that either do not affect or aid the victims switching speed should be allowed by the encoding algorithm.

The traditional approach to reducing the parasitic inductance within the package has been through aggressive package design. We are currently seeing success in the application of chip-scale and flip-chip technologies in level 1 interconnect for high-end applications. While such technologies decrease the above mentioned inductive effects, they are still relatively expensive for the majority of ICs. Further, they do not completely eliminate the inductive problems. Level 2 interconnect has been improved by moving toward surface mount and grid array style packaging. While these technologies are becoming affordable due to process improvements, they do not completely eliminate the inductance problem either. While aggressive package design assists in the problem, it is a slow and expensive process to develop new packages.

In this paper, we present a technique to reduce inductive crosstalk in the interconnect by eliminating the switching patterns being transmitted off-chip which induce noise above a user-specified limit. This is accomplished by inserting intermediate (stutter) states in the data transmission. These stutter states act as intermediate points between arbitrary bus vectors that would normally require a transition that resulted in a high level of switching noise. The stutter states are chosen such that transitions to and from the stutter states

3-9810801-0-6/DATE06 © 2006 EDAA

meet the noise requirements. These stutter states are ignored by the receiver but result in a reduced number of simultaneously switching signals in the off-chip data. This reduced number of switching signals translates into a higher per-pin datarate and increased bus performance. Our results show that the bus performance is increased even after accounting for the overhead due to the stutter states.

We begin by constructing a set of equations which encode the constraints that any legal transitions must satisfy to avoid supply bounce, signal glitching, and signal edge speed degradation. The degree of supply bounce, glitching and edge speed degradation that can be tolerated are expressed by means of user-specified parameters. From this set of equations, we construct a set of illegal transitions that the bus should avoid to bound switching noise to a user-defined level The set of illegal transitions is then used by the encoder to calculate the values and number of stutter states that must be inserted between vectors that originally resulted in an illegal transition. The output of the encoder is a transition map that shows how any vertex can transition to any other vertex without using an illegal transitions. In some cases, the transition may be direct. In other cases, the transition may require 1,2,..,$(2^{W_{segment}}-1)$ stutter states for a given bus segment size. The encoder guarantees that all transitions on the bus, including the stutter transitions, do not use any illegal transitions which results in a bounded amount of switching noise.

We show that the inter-chip bus throughput is increased as much as 225% for a 6-bit bus segment by using our encoding technique. The encoder overhead increases with bus segment width but is still shown to increase overall bus performance for noise limits of 5%, 10%, and 15% of V_{DD} for bus segments up to 8-bits wide. *The construction of the encoding algorithm is scalable to all sizes of off-chip busses by repeating identical, smaller encoded bus segments until the number of signal pins is reached.* Finally, stutter encoders of varying size and aggressiveness are implemented and synthesized using a TSMC 0.13um CMOS process. The functionality, size, and delay of the stutter circuitry demonstrates that this technique is suitable for incorporation into a modern VLSI design.

The rest of this paper is organized as follows. Section 2 provides the definitions used in the rest of this paper. Section 3 describes previous work on this topic. Section 4 provides the method used to translate switching noise into bus performance. Section 5 presents our stutter encoding scheme to reduce inductive cross-talk. Experimental results are presented in Section 6, and conclusions are drawn in Section 7.

2. Preliminaries and Terminology

Consider k segments of bus, with the j^{th} segment consisting of n signals $b_0^j, b_1^j, b_2^j \cdots b_{n-1}^j$. Let the vector sequence on segment j be denoted as v^j.

For example, if we had a V_{SS} and V_{DD} pin repeating after every 3 signal pins, the segments would consist of 5 pins. If the bus consisted of 15 signal pins, then we would implement it using 5 such segments.

- **Definition 1 :** *A Vector Sequence v^j is an assignment of values to the signals b_i^j as follows:*

  $b_i^j = v_i^j$, (where $0 \leq i \leq n-1$ and $v_i^j \in \{0, 1, -1\}$).

  Note that $v_i^j = 1(-1)$ indicates that the i^{th} signal of the j^{th} bus segment is rising (falling), while $v_i = 0$ indicates that it is either statically low or high.

- **Definition 2 :** *A Legal Vector Sequence (modulo inductive cross-talk) v is an assignment to the signals b_i such that:*

  - If b_i is a supply pin, the total bounce on this pin is bounded by P_{bnc} volts, where P_{bnc} is a user-specified constant.
  - if b_i is a signal pin which is static during the vector sequence, the glitch on this pin has a magnitude bounded by P_0 volts, where P_0 is a user-specified constant.
  - if b_i is a signal pin which is switching during the vector sequence, the switching speed of this pin is **not degraded** due to the effect of inductive cross-talk. Note that we can make this restriction stricter – by specifying that b_i's transition is in fact *sped up* due to inductive cross-talk.

Figure 1 shows an example of a bus using this terminology. In this figure, the bus consists of 3 ($k = 3$) bus segments ($j-1$, j, and $j+1$). We define the number of signals in any segment as W_{bus}. Each segment contains one V_{SS} pin and one V_{DD} pin giving the total size of any bus segment as $n = (W_{bus} + 2)$. Each pin has a self inductance of L_{11} and a mutual inductive coefficient k_i to the i^{th} neighboring pin. The transition on any pin is given by v_i^j which can take on a value of 0 (static), 1 (rising), or -1 (falling). Using this framework, any size bus can be represented by changing W_{bus} or adding additional segments. In addition, the transition of any pin (v_i^j) can be represented where signal pins can take on values of 0, 1, or -1 while V_{SS} and V_{DD} pins always take on a value of 0.

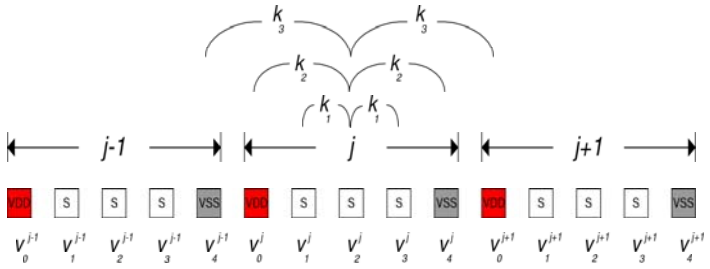

Figure 1: Example Bus Configuration

3. Previous Work

There has been much work into the reduction of parasitic inductance through package advancement [3, 6, 5]. Since the performance limitation is caused by the parasitic inductance in the level 1 and level 2 interconnects of the IC package, many packaging technologies have been developed to reduce this inductance. Table 1 shows the parasitic inductance values for three industry standard packages (a Quad FlatPak (QFP) with wire bonding, a Ball Grid Array (BGA) with wire bonding, and a flip-chip BGA package). In this table, L_{self} is the self-inductance of a pin, and the columns to its right are the mutual inductive coupling coefficients of successive neighbors of this pin. This table illustrates the evolution of package development over the past 10 years and the cost associated with moving toward packages that posses less inductive parasitics [7]. Since cost is the largest barrier to moving toward advanced packaging, any technique that can increase off-chip bus performance without changing the package is of considerable value [7].

Package	L_{self}	K_1	K_2	K_3	K_4	K_5	$Cost_{pin}$
QFP-wb	4.550nH	0.744	0.477	0.352	0.283	0.263	\$0.22
BGA-wb	3.766nH	0.537	0.169	0.123	0.097	0.078	\$0.34
BGA-fc	1.244nH	0.630	0.287	0.230	0.200	0.175	\$0.63

Table 1: Self and Mutual Inductance Values for Modern Packages

Techniques have also been presented to minimize the inductive problems due to packaging without changing the package. All of these techniques attempt to reduce the total amount of switching noise by reducing the instantaneous switching current. Pipeline damping was presented in [8] where the authors attempt to minimize peak current levels by using a multi-valued output driver. While this approach improves performance by reducing the inductive ringing, it requires complex circuitry to implement the multi-valued output

driver. In addition, while the occurrence of the maximum current spikes are reduced, they are not completely eliminated.

CODECs have also been presented [9] that limit the total number of simultaneously switching signals with the same transition direction. This has the effect of reducing the power supply bounce by limiting the total amount of current flowing through the power supply pins at any given time. This technique reported performance improvements but only considered the supply bounce and not the signal-to-signal cross-talk.

Switching noise reduction has also been attempted by simply delaying the edges of off-chip data relative to each other [10]. This technique reduces the peak currents pulled through the package instantaneously by making the edges occur at different points in time. This technique has the drawback that by delaying the edges relative to each other, the timing margin is reduced when using a synchronous or source synchronous clocking architecture. The reduced timing margin due to delayed edges will eventually limit the bus performance. A technique that can maintain tight timing relationships between edges and still reduce switching noise is desired to optimize the performance of the bus. In [11], a memory-based encoding scheme was reported, which avoided inductive cross-talk inducing patterns by growing the bus size. In contrast to this approach, our work avoids increasing the bus width, avoiding the noisy bus sequences by introducing intermediate stutter states.

Our work improves upon previous techniques by additionally considering signal rise-time degradation and glitching due to inductive cross-talk. In our approach, the maximum switching noise is bounded, which increases bus performance without changing packages or inserting channel-to-channel delay.

4. Performance Model

The amount of switching noise will directly effect the speed at which an off-chip bus can operate. The amount of noise that can be tolerated depends on the system but is usually between 5% to 15% of V_{DD} [16]. For this work we define arbitrary noise limits that bound the total amount of supply bounce (P_{bnc}), glitch magnitude (P_0), and edge coupling (P_1(rising) and P_{-1}(falling)). P_{bnc} is the amount of supply bounce noise in the system as a percentage of V_{DD} where $P_{bnc} \leq 1$ and the total amount of supply bounce is $P_{bnc} \cdot V_{DD}$. The total supply bounce noise can be expressed using equations 1 and 2. The supply bounce can also be bounded by setting the expression less than or equal to a user-defined noise limit, (i.e., $P_{bnc} \cdot V_{DD}$). Equation 3 gives the relationship between supply bounce noise and the user-defined noise limit P_{bnc}. In this expression, it is assumed that the worst-case switching pattern is present when evaluating the maximum noise in the system.

$$P_{bnc} \cdot V_{DD} \geq L_{11} \cdot W_{bus} \cdot (\frac{di}{dt}) + \sum M_{1k}(\frac{di}{dt}) \tag{3}$$

Similar expressions can be written for the signal coupling noise limits P_0, P_1, and P_{-1}.

$$P_0 \cdot V_{DD} \geq \sum M_{1k}(\frac{di}{dt}) \tag{4}$$

$$P_{1(-1)} \cdot V_{DD} \geq \sum M_{1k}(\frac{di}{dt}) \tag{5}$$

The summation in the equations above is for a fixed number of pins on either side of the pin under consideration. These expressions relate switching noise to the package parameters (Table I) and $\frac{di}{dt}$. To translate the switching noise to bus performance, we first define *slewrate* as:

$$slewrate = \frac{dv}{dt} = \frac{di}{dt} \cdot Z_{load} \tag{6}$$

where Z_{load} is the characteristic impedance of the transmission line being driven on the printed wiring board on which package is loaded. The rise time of the signal is defined as the time it takes to switch from 10% to 90% of the DC output value (80% of V_{DD}). This can be expressed in terms of *slewrate* by:

$$t_{rise} = \frac{0.8 \cdot V_{DD}}{slewrate} \tag{7}$$

The rise time can then be used to define the minimum Unit Interval (UI) that can used in a robust digital system [17, 18, 19]:

$$UI_{min} = (1.5) \cdot (t_{rise}) \tag{8}$$

The UI_{min} defines the minimum duration of the data valid window in order to transmit a logic symbol successfully. This corresponds to the maximum data rate of a signal as follows [17, 18, 19]:

$$DR_{max} = \frac{1}{UI_{min}} \tag{9}$$

The total system throughput TP of the bus can now be expressed as $TP = DR_{max} \cdot W_{bus}$. Using equations 3 through 9 we can express bus performance in terms of the user-defined parameters P_{bnc}, P_0, P_1, and P_{-1}. If P_{bnc} is the dominating noise source, then the maximum datarate is expressed as:

$$DR_{max} = \frac{P_{bnc} \cdot Z_{load}}{(1.5) \cdot (0.8) \cdot [L_{11} \cdot W_{bus} + \sum M_{1k}]} \tag{10}$$

If $P_{(0,1,-1)}$ is the dominating noise source, then the maximum datarate is expressed as:

$$DR_{max} = \frac{P_{(0,1,-1)} \cdot Z_{load}}{(1.5) \cdot (0.8) \cdot [\sum M_{1k}]} \tag{11}$$

In Equations 10 and 11, W_{bus} is used to represent the worst-case inductive switching pattern of the segment. This occurs when all of the signal pins within the segment transition in the same direction. When using these equations to predict the performance improvement of the encoding techniques, W_{bus} is changed to $W_{bus-eff}$ and represents the total number of signals that are allowed to simultaneously switch by the encoder where $W_{bus-eff} \leq W_{bus}$.

5. Our Approach

Our approach is to encode the off-chip data in a manner that eliminates the worst-case switching patterns prior to traversing the package. By eliminating the worst-case patterns, the maximum amount of noise that is present can be reduced. This translates into a faster datarate that can be achieved on the bus following Equations 10 and 11.

This is accomplished by first determining which transitions will not violate the user-defined noise limits P_{bnc}, P_0, P_1, or P_{-1}. A series of constraint equations are created that can be evaluated for each possible transition on the bus segment and indicate whether a transition results in a noise limit violation. If a transition results in a noise limit violation, then that transition is removed from the set of legal transitions that is allowed in the off-chip transmission.

Using the remaining legal transitions, a directed graph is created. The encoder algorithm is performed on the directed graph which finds the path between all vertices using only legal transitions. In the situation where two vertices cannot transition directly, an intermediate vertex is used to complete the transition. By using an intermediate vertex, the bus can switch between the two original vertices using only legal transitions within the directed graph. The intermediate vertex is called a *stutter state* and is ignored by the receiver by gating out the source synchronous clock when the stutter state is transmitted. Multiple stutter states can be used depending on how aggressively the segment is constrained. This algorithm is performed on a representative segment of the bus which is typically much smaller than the entire bus.

5.1 Constraint Equations

The first step in creating the stutter encoder is to create a set of constraint equations. The constraint equations are written so that arbitrary transitions can be evaluated for noise limit violations. When a transition is evaluated using the constraint equations and violates one of the user-defined noise limits, the transition is flagged as *illegal* and is removed from set of transitions that are allowed to be driven through the package interconnect. Each of the possible off-chip transitions are evaluated in each of the constraint equations. After the evaluation is complete, a subset of *legal* transitions remain which are used in the construction of the directed graph.

5.1.1 Supply Bounce Constraints

When a pin i in segment j is a V_{DD} or V_{SS} pin, it is required that the bounce magnitude due to the electrical parasitics in the package must not exceed the user-defined noise limit P_{gnd}. Since the package noise occurs only when transitions are present, the constraint evaluations are performed on the transition values on the bus (i.e., $v_i^j = 0/1/-1$). The constraint equation takes into account the voltage noise due to the self-inductance of the supply pin in addition to any mutual inductive coupling that occurs due to switching signals in adjacent pins. By multiplying the coupling magnitude by the transition value v_i^j, the polarity of the transition is accounted for. This handles the situations in which a static signal pin (v_i^j=0) has no coupling effect in addition to the cumulative nature of coupling as multiple signal pins switch. The following constraint equation is written for any pin within a bus segment j that is used for V_{DD} or V_{SS} and is being evaluated for a supply bounce violation:

- $v_i^j = V_{DD}$ *or* $V_{SS} \Rightarrow$
  $P_{gnd} \cdot V_{DD} \geq (\frac{di}{dt}) \cdot [(L_{11}) \cdot (N_{1/-1}) + \sum[(M_{1(|k|+1)}) \cdot (v_{i+k}^j)]]$

In this constraint, $N_{1/-1}$ represents the total number of signals that are rising (falling) in the bus segment for any given constraint evaluation. When the pin under evaluation is a V_{DD} pin, N_1 is used to represent the worst-case supply bounce situation when signals are transitioning from a logic 0 to a logic 1. When the pin under evaluation is a V_{SS} pin, N_{-1} is used to represent the worst-case ground bounce situation when signals are transitioning from a logic 1 to a logic 0. The summation of the mutually coupled voltage is evaluated over the range of pins that have a significant coupling magnitude. Typically, signals with coupling coefficients less than 0.15 are ignored (Table 1) which reduces the computation time.

5.1.2 Signal Coupling Constraints

When a pin i in segment j is a signal pin, it is required that the coupled voltage onto that pin does not exceed any of the user-defined noise limits for signal coupling. If the signal pin is static (v_i^j=0), then the glitch magnitude onto the victim pin must not exceed P_0. As in the constraint equations for supply bounce, the coupling contribution is multiplied by the transition value v_i^j to account for the polarity and cumulative effect of coupling due to switching neighbors. The following constraint equation is written for any signal pin within a bus segment j that is static (v_i^j=0) and being evaluated for a glitch violation:

- $v_i^j = 0 \Rightarrow$
  $P_0 \cdot V_{DD} \geq (\frac{di}{dt}) \cdot [\sum[(M_{1(|k|+1)}) \cdot (v_{i+k}^j)]]$

When a signal pin i in segment j is transitioning from a logic 0 to a logic 1 (v_i^j=1), it is required that the coupled voltage onto that pin does not hinder the risetime. In this situation, the cumulative nature of the mutual coupling can be exploited to actually aid the transition on the victim signal pin. By requiring that the cumulative coupling voltage on the victim pin either matches or exceeds the user-defined noise limit P_1, it is also required that the victim risetime is either unhindered or improved (i.e., sped up). The following constraint equation is written for any signal pin within a bus segment j that is undergoing a positive transition (v_i^j=1) and being evaluated for a rising edge degradation violation:

- $v_i^j = 1 \Rightarrow$
  $P_1 \cdot V_{DD} \leq (\frac{di}{dt}) \cdot [\sum[(M_{1(|k|+1)}) \cdot (v_{i+k}^j)]]$

In a similar manner, when a signal pin i in segment j is transitioning from a logic 1 to a logic 0 (v_i^j=-1), it is required that the coupled voltage onto that pin does not hinder the falltime. By requiring that the cumulative coupling voltage on the victim pin either matches or exceeds the user-defined noise limit P_{-1}, it is also required that the victim falltime is either unhindered or improved. The following constraint equation is written for any signal pin within a bus segment j that is undergoing a negative transition (v_i^j=-1) and being evaluated for a falling edge degradation violation:

- $v_i^j = -1 \Rightarrow$
  $P_{-1} \cdot V_{DD} \leq (\frac{di}{dt}) \cdot [\sum[(M_{1(|k|+1)}) \cdot (v_{i+k}^j)]]$

5.2 Constructing the Encoder

Transitions which result in a user-defined noise limit violation are removed from the subset of legal transitions. The remaining legal transitions are used to create the directed graph G which represents all of the legal paths between any two vertices. The graph is represented implicitly and efficiently using ROBDDs [20, 21].

The stutter encoder is constructed by evaluating each vertex v_s of $G(V, E)$ and finding the shortest path between v_s and any destination vertex v_d in G using only legal edges of G (including self-edges). In order to be able to construct a stutter encoder, the following conditions must hold:

- There must exist at least two outgoing edges (including the self-edge) for each $v_s \in G$.
- There must exist at least two incoming edges (including the self-edge) for each $v_d \in G$.

These requirements ensure that for the directed graph G, each vertex can reach at least one other vertex and can be reached by at least one other vertex. If these requirements are not met, then the user must relax the user-defined noise limits until both conditions are met.

Given G, the algorithm first tests whether both of the above mentioned requirements are satisfied. The algorithm next attempts to determine the number of intermediate steps required for a vertex $v_s \in G$ to reach another vertex $v_d \in G$. If v_d can be reached with just one edge, the algorithm records the transition as a *direct* transition (one that requires 0 stutter steps).

For the case where v_s can *not* reach v_d with only one edge, then at least one stutter state is needed to complete the transition. The algorithm then attempts to find a path between v_s and v_d using two edges. Since the set of vertices V_d that can be reached from v_d by means of a direct path is known, the algorithm simply needs to find an edge from v_s to $v \in V_d$. Once such a path between v_s and v_d is found, then the algorithm records the intermediate vertex as a necessary *stutter state* which is required between v_s and v_d. This process is repeated for transitions which require more stutter states.

Algorithm 1 contains the pseudo-code for the stutter encoder algorithm. All *transition_path* variables are initially initialized to be empty. The routine $find\_path(v_s, v_d, l)$ returns a shortest path from the source v_s to destination v_d, with path_length l.

The maximum possible number of stutter states that may be used is $(2^{W_{bus}} - 1)$. This represents the worst-case where in order to transition from v_s to v_d, each and every other vertex within G must be used as a stutter state. While this represents the absolute worst case, experimental results have shown that the number of stutter states is typically between 0 and 3 for bus segments up to 8 bits. It should be noted that this analysis is only performed on a representative bus segment (which is typically very small compared to the entire off-chip bus).

Algorithm 1 Constructing the Stutter Encoder

for $(each\ v_s \in V)$ **do**
  for $(each\ v_c \in V)$ **do**
    for $(path\_length = 1;\ path\_length < 2^{W_{bus}-1};\ path\_length++)$ **do**
      if $(transition\_path[v_s, v_d] == \phi)$ **then**
        $transition\_path[v_s, v_d] = find\_path(v_s, v_d, path\_length)$
      end if
    end for
  end for
end for
$find\_edge(v_s, v_d, l)$
if $(\exists v_1, v_2, \cdots v_{(l-1)} S.T.((v_s, v_1) \in E) \wedge ((v_2, v_2) \in E) \wedge \cdots \wedge ((v_{(l-1)}, v_d) \in E))$ **then**
  $return\ (v_s, v_1, v_2, \cdots, v_{(l-1)}, v_d)$
else
  $return\ \phi$
end if

5.3 Encoder Overhead

To calculate the overhead of the stutter encoder, it is assumed that each vector $v_s \subseteq V$ has an equal probability of occurring on the bus. Using this assumption, a sequence of data patterns is constructed, in which each and every sequence occurs on the bus at least once. When this sequence is transmitted, the minimum number of stutter states are inserted in the transition between any pair of vectors. The maximum number of stutter states that will be inserted in a sequence for any given encoder is $2^{W_{bus}-1}$. Equation 12 gives the overhead of the stutter encoder.

$$Overhead = \left(\frac{\sum_{k=1}^{2^{(W_{bus}-1)}} (\#\ Trans\ with\ k\ Stutters) \cdot k}{2^{(2 \cdot W_{bus})}}\right) \quad (12)$$

5.4 Decoder Construction

The stutter encoding technique assumes a source synchronous clocking architecture. In source synchronous clocking, the bus clock is generated at the transmitter and synchronized to the off-chip data being transmitted. The clock is then transmitted along with the data in the off-chip bus. By doing this, the timing correlation between the clock and data is extremely tight. This architecture has seen wide adoption in industry as a way to address channel-to-channel skew and common mode noise.

The stutter encoding technique is specifically designed for a source synchronous architecture. The encoding circuitry gates out the source synchronous clock when stutter states are transmitted off-chip. Since the receiving circuitry only acquires data on the rising edge of the source synchronous clock, the stutter states are ignored. In this manner, no special decoding circuitry is needed.

6. Experimental Results

To verify the feasibility of this encoding technique, experimental results where performed the BGA, wire-bonded package listed in table 1. For this package, a fixed $\frac{di}{dt}$ of 8 MA/s was used which corresponds to a one channel data rate of 222Mb/s using Equation 10. As channels are added to the bus segment, the amount of switching noise increases. Figure 2 shows how the supply bounce noise increases as signals are added to the segment following Equations 1 and 2. For these experiments, three noise limits were chosen (5%, 10%, and 15% of V_{DD}). Figure 2 illustrates that as channels are added to the bus segment, these noise limits are violated. In order to meet these noise limit requirements in an un-encoded system, the per-pin datarate needs to be decreased as channels are added.

Segment sizes from 2 to 8 signal pins were encoded with our stuttering technique using the three sets of noise limits. In the 5% noise limit, the user-defined noise parameters P_{bnc}, P_0, P_1, and P_{-1} were all set to 5% of V_{DD}. Similarly for the 10% and 15% noise limits. The off-chip data was encoded such that the 8 MA/s was not decreased and no vector sequences were allowed that caused the noise to exceed the user-defined limits. For these conditions, constraint equations were written and evaluated for all possible transitions in the bus segment. From the constraint evaluations, the subsets of legal transitions were found for each of the noise limit conditions.

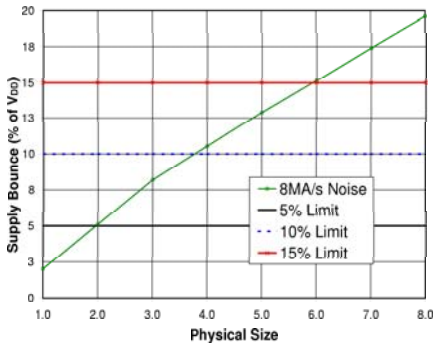

Figure 2: Supply Noise Versus Bus Size for Un-encoded 8 MA/s Bus

These legal transitions where then used in the creation of directed graphs which were evaluated using algorithm 1. Figure 3 shows the overhead of the encoder designs. Table 2 lists the percentage of transitions that require stutter states for each of the encoders.

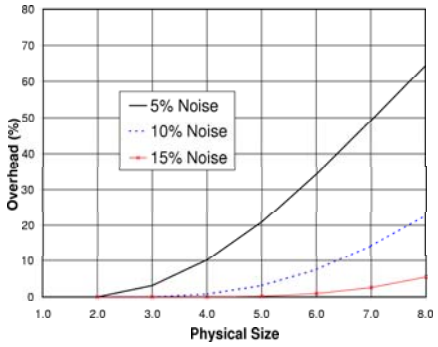

Figure 3: Overhead of the Stutter Encoding Technique

The stutter encoder ensures that the worst-case bus patterns are never transmitted off-chip. This bounds the total number of simultaneously switching signals in the segment to $W_{bus-eff}$. $W_{bus-eff}$ is then used in Equations 10 and 11 to calculate the improved datarate and throughput of the encoded data. The net improvement must account also account the overhead of the encoder. Figure 4 shows the bus performance improvement when using the stutter encoder technique. The improvement is in terms of the percentage increase in throughput of the bus after including the encoder overhead. The improvement of the 5% encoder reaches a performance maximum of 225% at a segment size of 6-bits. After this, the overhead of the stutter encoder begins to outweigh the increased per-pin datarate. However, all three noise limits experience positive improvement up to bus segment sizes of 8-bits even after considering the encoder overhead.

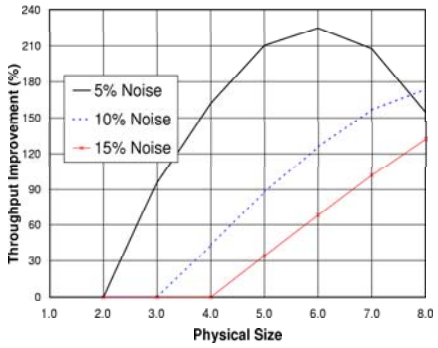

Figure 4: Bus Performance Improvement Due To Encoder

Noise Limit	Bus Size	Number of Stutter Steps			
-	-	0	1	2	3
5% Limit	2	100	0	0	0
	3	96.9	3.1	0	0
	4	89.8	10.2	0	0
	5	79.3	20.5	0.2	0
	6	66.6	32.5	0.9	0
	7	53.4	44	2.6	0
	8	41.1	53.4	5.4	0.1
10% Limit	2	100	0	0	0
	3	100	0	0	0
	4	99.2	0.8	0	0
	5	96.9	3.1	0	0
	6	92.5	7.5	0	0
	7	85.9	14.1	0	0
	8	77.3	22.6	0.1	0
15% Limit	2	100	0	0	0
	3	100	0	0	0
	4	100	0	0	0
	5	99.8	0.2	0	0
	6	99.1	0.9	0	0
	7	97.4	2.6	0	0
	8	94.5	5.5	0	0

Table 2: Percentage of Transitions Requiring Stutter States

The encoders where then implemented using the Verilog HDL and synthesized in a TSMC 0.13um CMOS process. The implementation consists of a pipeline in which each stage of the pipe was routed to a multiplexer which drives the bus patterns off-chip. A state machine monitors the incoming data from the core of the IC to check whether a stutter state is needed in the off-chip transmission. At the beginning of circuit operation, the output of the first pipeline stage is selected to be output of the multiplexer. When a sequence of vectors occurs which require a stutter state, then the multiplexer is switched to the state machine input where the appropriate stutter state is output. After the stutter state(s) is output, then the state machine switches the multiplexer back to the pipeline but now selects the next stage of the pipe. The state machine continues to monitor the pipeline for illegal adjacent states and inserts the appropriate number of stutter states in the off-chip data transmission. During the time in which a stutter state is being selected as the output of the multiplexer, the control state machine gates out the source synchronous clock. By gating out the clock, this insures that the receiver will not latch in any of the stutter states. Figure 5 shows the schematic for the stutter encoder circuit.

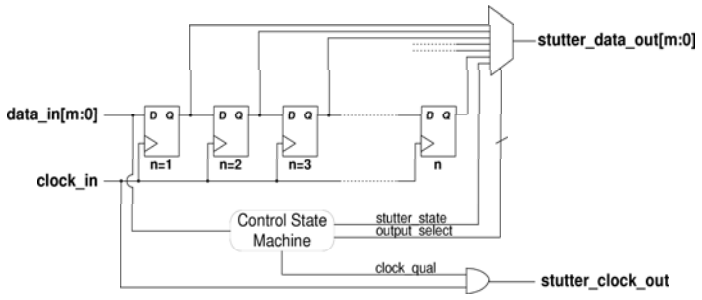

Figure 5: Bus Stuttering Encoder Schematic

Tables 3 lists the delay and area required to implement the stutter encoders in a TSMC 0.13um CMOS process. The combinational delay in this table is left unoptimized to illustrate the relative differences between encoder designs. However, this delay can easily be hidden using advanced architectural techniques such as pipelining of the combinational delay. The area of these encoders is shown to be less than 1.5% for a $5mm^2$ die size. These results illustrate that the circuitry required for the encoders is fast and small enough to be easily implemented in a modern CMOS process.

	Bus Size	Noise Limit		
	-	5%	10%	15%
Delay (ns)	4	2.02	1.99	1.90
	6	2.42	2.38	2.30
	8	2.85	2.79	2.69
Area (um^2)	4	311k	310k	239k
	6	362k	345k	341k
	8	389k	368k	359k

Table 3: Stutter Encoder Synthesis Results in a TSMC $0.13um$ Process

7. Conclusions

Simultaneous switching noise in IC packaging is one the limiting factors to performance in modern systems. This issue has traditionally been handled through aggressive package design. However, package design if often too slow and expensive for the majority of applications.

In this work, we presented an encoding technique for off-chip data transmission that will reduce switching noise by avoiding worst-case transitions. By reducing the switching noise, the per-pin datarate of the bus can be increased and the overall throughput can be improved even after considering the overhead of the encoder. The encoding technique involves inserting intermediate (stutter) states in the data sequence to ensure that the worst-case noise patterns are never used. The number of stutter states inserted depends on how aggressively the user-defined noise limits are chosen.

Experimental results demonstrate that the encoding technique improves bus performance for all segment sizes studied. For the 5% noise limit, the stutter encoder achieved the greatest improvement at a segment size of 6-bits with a 225% increase in throughput compared to an un-encoded segment of the same size. Synthesis is a TSMC 0.13um process indicated that the area and delay of the encoders are negligible and can easily be implemented in a modern VLSI process.

References

[1] "The International Technology Roadmap for Semiconductors." http://public.itrs.net, 2003.

[2] R. Tummalo, *Fundamentals of Microsystem Packaging*. McGraw-Hill, 2001.

[3] M. Miura, N. Hirano, Y. Hiruta, and T. Sudo, "Electrical characterization and modeling of simultaneous switching noise for leadframe packages," in *Proceedings of 45th Electronic Components and Technology Conference*, pp. 857–864, May 1995.

[4] B. Young, "Return path inductance in measurements of package inductance matrixes," in *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, vol. 20, Feb 1997.

[5] N. Hirano, M. Miura, Y. Hiruta, and T. Sudo, "Characterization and reduction of simultaneous switching noise for a multilayer package,"

[6] M. Lopez, J. Prince, and A. Cangellaris, "Influence of a floating plane on effective ground plane inductance in multilayer and coplanar packages," in *IEEE Transactions on Advanced Packaging*, vol. 22, pp. 182–188, May 1999.

[7] Agilent Packaging Group, Ft. Collings, CO. Personal Communication, 2004.

[8] M. Powell and T. Vijaykumar, "Pipeline damping: a microarchitectural technique to reduce inductive noise in supply voltage," in *Proceedings of 30th International Symposium on Computer Architecture*, pp. 72–83, June 2003.

[9] C. Chen and B. Curran, "Switching codes for delta-i noise reduction," in *IEEE Transactions of the 43rd IEEE Midwest Symposium on Circuits and Systems*, vol. 45, pp. 1017 – 1021, Sept 1996.

[10] Actel Inc., "Application Note: Simultaneous Switching Noise and Signal Integrity." www.actel.com.

[11] B. LaMeres and S. Khatri, "Encoding-based minimization of inductive cross-talk for off-chip data transmission," in *Proceedings. Design, Automation and Test in Europe (DATE) Conference*, vol. 2, pp. 1318–1323, 2005.

[12] C. Duan and S. Khatri, "Exploiting crosstalk to speed up on-chip buses," *Design Automation and Test in Europe Conference*, Feb 2004.

[13] B. Victor and K. Keutzer, "Bus encoding to prevent crosstalk delay," in *Proceedings. IEEE/ACM International Conference on Computer Aided Design*, (San Jose, CA), pp. 57–63, Nov 2001.

[14] C. Duan, A. Tirumala, and S. Khatri, "Analysis and avoidance of cross-talk in on-chip buses," *IEEE Symposium on High-Performance Interconnects (HOT Interconnects)*, pp. 133–138, Aug 2001.

[15] E. Mejia-Motta, F. Sandoval-Ibarra, and J. Santana, "Design of cmos buffers using the settling time of the ground bounce voltage as a key parameter," in *Proceedings of 43rd IEEE Midwest Symposium on Circuits and Systems*, vol. 2, pp. 718–721, Aug 2000.

[16] M. Horowitz, C. Yang, and S. Sidiropoulos, "High-Speed Electrical Signaling: Overview and Limitations," *IEEE Micro.*, vol. 18, pp. 12–24, Jan 1998.

[17] C. Kinnaird, "Standards are key to optimizing high-speed data bus communications," *Planet Analog (planetanalog.com)*, Oct 2002.

[18] H. Johnson and M. Graham, *High-Speed Signal Propagation*. Prentice Hall PTR, 2003.

[19] H. Johnson and M. Graham, *High-Speed Digital Design*. Prentice Hall PTR, 2003.

[20] R. E. Bryant, "Graph based algorithms for Boolean function representation," in *IEEE Transactions on Computers*, vol. C-35, pp. 677–690, August 1990.

[21] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," in *Design Automation Conference (DAC)*, pp. 40–45, June 1990.