Variable Frequency Crosstalk Noise Analysis: A Methodology to Guarantee Functionality from dc to f_{max}

Byron Krauter
IBM Corporation
Austin, TX 78758-3493
krauter@us.ibm.com

David Widiger
IBM Corporation
Austin, TX 78758-3493
widiger@us.ibm.com

Abstract - One means of reducing pessimism in crosstalk analysis is to consider timing orthogonality. While earlier works have addressed the temporal alignment of timing windows [1, 2, 3, 4], these treatments have overlooked one key point. Crosstalk noise failures are frequency dependent. A chip that functions at one frequency can fail due to crosstalk noise at faster and slower frequencies. Moreover, because system developers and manufacturers need chips that operate over a wide range of frequencies, noise analysis tools should guarantee a wide range of operation. In this paper, we explain this phenomenon and propose a simple procedure to include timing information and guarantee functionality from dc to f_{max}.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids - *simulation, verification*

General Terms

Algorithms, Design, Verification

Keywords

crosstalk, noise analysis, timing windows, timing orthogonality, GCD frequency, LCM window, frequency-dependent noise

1. Introduction

While it is paramount that a crosstalk noise analysis tool should never under-predict noise, it is also important that these tools not be too pessimistic since undue pessimism can produce an uncompetitive product (either a larger die size or a schedule slip fixing false failures). One way to reduce pessimism in crosstalk noise analysis is to consider timing windows and timing orthogonality.

Timing windows come in two flavors, aggressor switching and victim sensitivity windows. Aggressor switching windows define when aggressor nets can switch, and victim sensitivity windows define when victim pins are sensitive to noise. Aggressor switching windows are specified at source pins, and range from the earliest to latest source pin arrival times. (The aggressor source to victim sink pin noise delays are calculated during noise analysis.)

Victim sensitivity windows are specified at the victim sink pin, and extend from the pin set-up to the pin hold time. While the concepts of set-up and hold time are commonly understood for clocked circuits - clock gates, latches, and dynamic circuits - these concepts also apply to non-clocked circuits. The set-up and hold times of a non-clocked circuit are found by backwards propagating clocked circuit set-up and hold times back to the victim sink pin. To be more specific, by subtracting the long and short path delays from the clocked circuit set-up and hold times respectively.

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

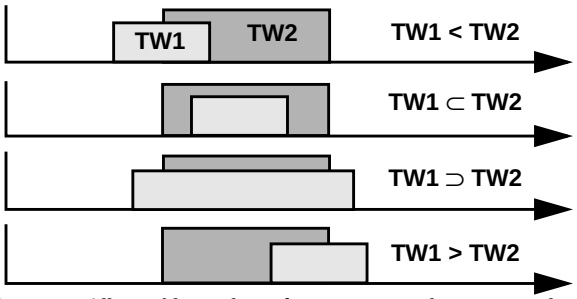

FIGURE 1: All possible overlaps of two timing windows TW1 and TW2

For single frequency operation, the recipe for examining timing orthogonality is straightforward. Timing windows are gleaned from timing analysis results, and two windows overlap when their minimum and maximum edges satisfy the following inequalities:

$$TW1_{min} < TW2_{max} \quad (1)$$

and

$$TW2_{min} < TW1_{max}. \quad (2)$$

In Fig. 1, we've depicted every possible overlap of timing windows TW1 and TW2, and indeed, every case satisfies (1) and (2).

An accurate noise analysis at a single frequency begins by considering only those aggressor nets whose noise pulse windows overlap the victim sensitivity window. Accuracy can be further enhanced by not superimposing the individual peak noises but instead by reanalyzing the coupled network with a nonlinear victim driver model when all the aggressors whose noise pulse windows overlap the victim sensitivity window, switch simultaneously. [1] describes such a procedure as well as how to include the effects of propagated noise from the previous stage.

However, chips are not run at a single frequency. Systems are brought up and debugged at many frequencies. Manufacturing test may require functionality over a range of frequencies. And finally, the final product frequency may be different than the design target frequency. While the requirement to operate at slower frequencies doesn't affect timing analysis, it does affect noise analysis. That is, a chip that meets timing requirements at f_{max} will also meet timing requirements at slower frequencies, but a chip that operates at f_{max} can fail at a slower frequency due to noise.

2. Why Frequency Affects Noise Analysis

Timing window edges cannot be assigned to absolute times. An individual timing window edge is temporally fixed to one (or more) clock edges. Crosstalk noise is frequency dependent because the temporal position of two timing window edges fixed to different clock edges, changes with frequency. The following paragraphs demonstrate this phenomenon with a few examples.

2.1 Single Clock Example

Even the simplest designs - static CMOS chips using Master-Slave latches and a single clock - exhibit this frequency-dependent behavior. Windows that don't overlap at one frequency can overlap when the chip is run either faster or slower. Consider the example depicted in Fig. 2. Both the aggressor launch and the victim cap-

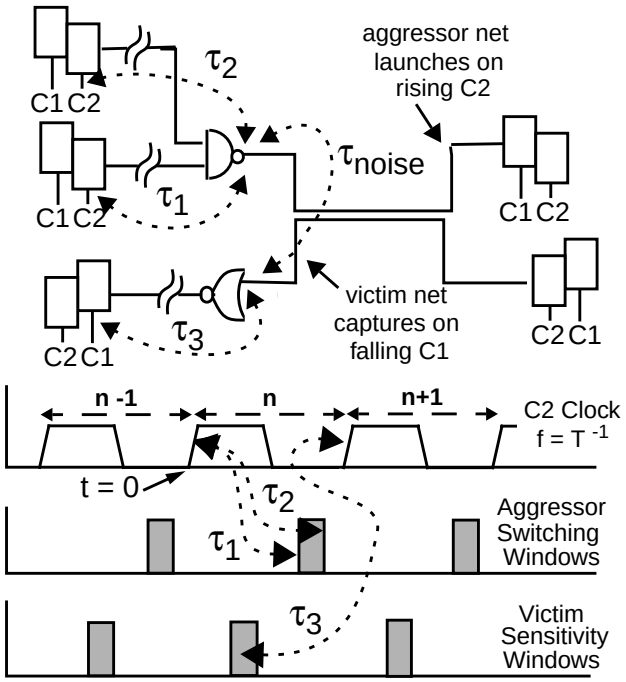

FIGURE 2: Changing Temporal Alignment of Aggressor Switching and Victim Sensitivity Window in a Single Clock Edge Design

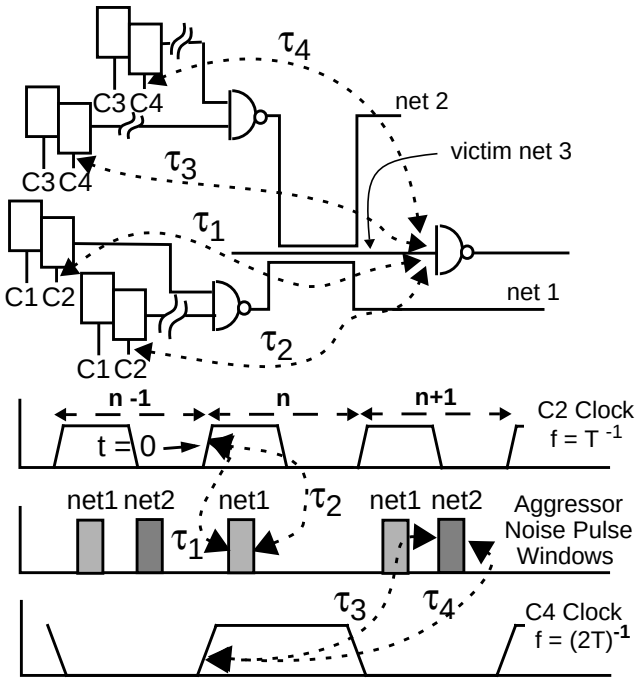

FIGURE 3: Changing Temporal Alignment of Aggressor Switching Windows with Clock Frequencies f and f/2.

ture are triggered on a rising C2 (or falling C1) clock edge. For convenience t=0 has been assigned to the rising C2 clock edge in the n*th* cycle. At the clock frequency depicted, the aggressor switching window and victim sensitivity window do not overlap.

In the n*th* cycle, the aggressor switching window at the nand gate output ranges from τ_1 to τ_2. When noise can appear at the nor gate input then ranges from ($\tau_1 + \tau_{noise}$) to ($\tau_2 + \tau_{noise}$) due to the noise pulse delay τ_{noise}. And finally, the victim sensitivity window at the nor gate input ranges from ($T - \tau_{set\text{-}up} - \tau_3$) to ($T + \tau_{hold} - \tau_3$) where $\tau_{set\text{-}up}$ and τ_{hold} are the setup and hold times at the downstream latch. To summarize

$$\tau_1 < \begin{matrix}\text{Switching Window}\\ \text{at Nand Gate Output}\end{matrix} < \tau_2, \quad (3)$$

$$\tau_1 + \tau_{noise} < \begin{matrix}\text{Noise Pulse}\\ \text{Window at Nor}\\ \text{Gate Input}\end{matrix} < \tau_2 + \tau_{noise}, \quad (4)$$

and

$$T - \tau_{setup} - \tau_3 < \begin{matrix}\text{Victim}\\ \text{Sensitivity}\\ \text{Window}\end{matrix} < T + \tau_{hold} - \tau_3. \quad (5)$$

Applying (1) and (2) to the windows defined by (4) and (5), yields a range of cycle times where the two windows will overlap.

$$\tau_1 + \tau_{noise} - \tau_{hold} + \tau_3 < T < \tau_2 + \tau_{noise} + \tau_{setup} + \tau_3 \quad (6)$$

So the non-overlap in Fig. 2 is a consequence of the frequency depicted. If the period T is increased to satisfy (6), the windows will overlap, potentially causing a noise failure at a slower frequency.

2.2 Multiple Synchronous Clocks Example

In the previously-discussed example where all signals are launched and captured on the same clock edge, frequency dependent crosstalk failures result from the relative movement between aggressor switching and victim sensitivity windows. With multiple synchronous clocks, frequency dependent failures can also result from the relative movement of switching windows when aggressor nets are launched on different, yet synchronous, clock frequencies.

Consider the simple example in Fig. 3. Nets 1 and 2 are two synchronous aggressors nets that couple to a third victim net 3. (For the purpose of this discussion, victim net 3 need not be synchronous to nets 1 and 2.) Net 1 is launched on a rising C2 clock running at a frequency f, and net 2 is launched on a rising C4 clock running at a frequency f/2.

At the frequencies depicted in Fig. 3, the (aggressor clock pin to victim sink pin) noise pulse delays τ_1 and τ_2 on net 1 and τ_3 and τ_4 on net 2 disallow overlap of the noise pulse windows. But when the common source clock frequency f is decreased, the overlap can and will occur every other cycle of the C2 clock.

Net 1 can produce a noise pulse at the victim sink every C2 clock cycle. In cycle n, the net 1 noise pulse window is given by

$$\tau_1 < \begin{matrix}\text{Net 1 Noise Pulse}\\ \text{Window in Cycle n}\end{matrix} < \tau_2 \quad (7)$$

and in cycle n + 1, the window is

$$\tau_1 + T < \begin{matrix}\text{Net 1 Noise Pulse}\\ \text{Window in Cycle n+1}\end{matrix} < \tau_2 + T. \quad (8)$$

Net 2 produces a noise pulse at the victim sink every other cycle of the C2 clock. In cycle n of the C2 clock, the noise pulse window is

$$\tau_3 < \begin{matrix}\text{Net 2 Noise Pulse}\\ \text{Window in Cycle n}\end{matrix} < \tau_4 . \quad (9)$$

While the windows defined by (7) and (9) will never overlap, applying (1) and (2) to the timing windows defined by (8) and (9) yields a cycle time range where the noise pulse windows overlap:

$$\tau_3 - \tau_2 < T < \tau_4 - \tau_1 \quad (10)$$

So while aggressors nets 1 and 2 do not simultaneously couple onto victim net 3 at the frequency depicted in Fig. 3, simultaneous coupling will occur at a slower frequencies. And again, this overlap could potentially cause a noise failure at a slower frequency.

3. Crosstalk Noise Analysis from dc to f_{max}

The following paragraphs will explain the general analysis procedure by first discussing timing windows and their relationship with clock edges, introducing new concepts that are critical to the

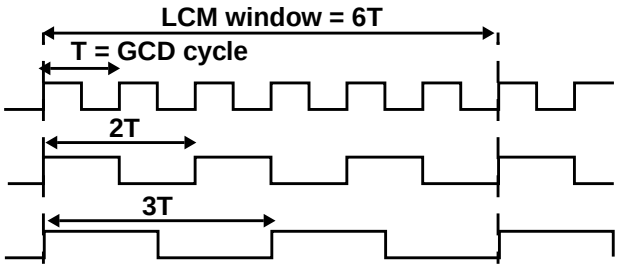

FIGURE 4: Least Common Multiple and Greatest Common Divisor Cycle Times for Synchronous Clocks with T, 2T, and 3T Cycle Times

analysis, and then discussing more specific analysis procedures that naturally build up to the general case. This section will conclude with a discussion of the more general analysis procedure that handles both asynchronous and synchronous nets whose frequencies range from dc to f_{max}.

3.1 Timing Windows and Clock Edges

Timing window edges are controlled by clock edges. The timing windows presented in Fig. 2 and 3 were single-clock-edge windows. That is, the leading and trailing edges were controlled by the same clock edge. In general, static logic circuits and master-slave latches produce single-clock-edge windows, and dynamic circuits, clock gates, and more complicated latches create windows that are controlled by multiple clock edges.

Transparent latches, for example, can create timing windows where the controlling clock edges change with frequency. At high frequencies when data flushes through the latch, leading and trailing edges are controlled by different clock edges. But at slower frequencies when the latch relaunches every signal, both edges are controlled by the same clock edge. Other timing window subtleties that aren't apparent in Fig. 2 and 3 include: When a upstream gate logically combines signals launched from different clocks, the downstream aggressor nets have multiple timing windows. When a victim sink feeds multiple downstream latches that are clocked with different, but synchronous clocks, the victim sink has multiple sensitivity windows.

3.2 Least Common Multiple (LCM) and Greatest Common Divisor (GCD) Cycle Times

For a group of synchronous clocks, the least common multiple and the greatest common divisor cycle times, T_{lcm} and T_{gcd}, are the cycle times of the beat and "source" clocks. Consider the three clocks depicted in Fig. 4. With periods of T, 2T, and 3T, the relative position of these clocks repeats every $T_{lcm} = 6T$, the least common multiple of 1, 2, and 3. And every clock frequency can be expressed as an integer multiple of $T_{gcd} = T$, the greatest common divisor of 1, 2, and 3. Moreover, every clock edge can also be expressed as a half integer multiples of T_{gcd}.

The importance of T_{lcm} and T_{gcd} is that all possible alignments between any two timing windows can be found inside an arbitrary LCM window, and every timing window inside the LCM window can expressed as a linear function of the GCD cycle time, T_{gcd}. Because every edge is temporally pinned to one (or more) GCD clock edges, these expressions are of the form

$$\tau_{edge} = \frac{n}{2} T_{gcd} + \tau_{offset} \qquad (11)$$

where $0 \le n < 2T_{lcm} / T_{gcd}$.

3.3 Fixed Frequency Analysis with Multiple, Yet Synchronous, Clocks

We now describe here a new procedure for examining all possible window alignments that can occur among signals derived from multiple, yet synchronous, fixed-frequency clocks. This procedure will make use of the previously-discussed LCM window concept.

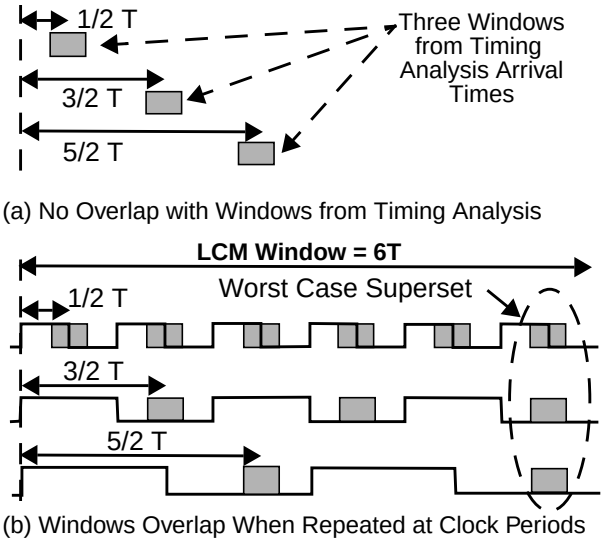

FIGURE 5: Three Aggressor Switching Windows Clocked at T, 2T, and 3T and How They Repeat Inside Their 6T LCM Window

Consider a group of aggressor nets and a victim sink that are synchronous but do not all share the same clock period. We would begin a fixed frequency noise analysis by obtaining, from timing analysis, switching windows for every aggressor net, sensitivity windows for the victim sink, and the clock frequencies at which these windows repeat. Because we are considering fixed clock frequencies, timing window edges could be treated as absolute times, and not functions of T_{gcd} like that shown in (11).

Next we would determine via circuit simulation, the noise waveforms caused by each aggressor net switching by itself. The so-obtained noise pulse delays would be added to the aggressor switching windows to obtain aggressor noise pulse windows.

We would then determine the group LCM cycle time and build a list of all timing windows inside an arbitrary LCM window. This means repeating every timing window at its own fixed clock frequency across the LCM window. Timing windows that overlapped the LCM window endpoints would be split, and those portions outside the LCM window would be discarded.

Every entry in the list would have a window type (aggressor noise pulse or victim sensitivity), a net or pin identity, and the timing window endpoints. After sorting the list by timing window start times, we would traverse the list to determine which windows were simultaneously active. And in particular, we would determine via superposition, the set of noise pulse windows overlapping the sensitivity window that would result in the worst noise. Accuracy could be further enhanced by re-analyzing the coupled network with a nonlinear victim driver model when all the aggressors identified in the previous step, switch simultaneously [1].

The need for a LCM window analysis can be demonstrated with an example. Imagine timing analysis reported the three windows centered about 1/2, 3/2, and 5/2 T in Fig 5-a. When viewed as single windows, these windows obviously do not overlap. But if these windows were repeated at the clocks periods T, 2T, and 3T shown in Fig 5-b, every possible combination of timing window overlap would occur over the LCM window of 6T.

3.4 Variable Frequency Analysis with Multiple, Yet Synchronous Clocks

For the same set of synchronous aggressor nets and victim sink, a variable frequency noise analysis would also determine which

clock edges controlled the leading and trailing timing window edges, and express each edge as a function of the group GCD cycle time, T_{gcd}. The analysis would then determine the GCD clock ranges where pairs of timing windows overlapped. This determination can be best described as a double nested `for` loop that examines (1) and (2) for every pair of timing windows and finds a range of T_{gcd}, if one exists, that satisfies both inequalities.

For example, given a timing window TW1 with a leading edge

$$TW1_{min} = \frac{n_1}{2} T_{gcd} + \tau_1 \quad (12)$$

and another timing window TW2 with a trailing edge

$$TW2_{max} = \frac{n_2}{2} T_{gcd} + \tau_2, \quad (13)$$

a test of (1) ($TW1_{min} < TW2_{max}$) above yields

$$T_{gcd} < \frac{\tau_2 - \tau_1}{n_1 - n_2}. \quad (14)$$

If this and the subsequent test of (2) ($TW2_{min} < TW1_{max}$) both yield a GCD cycle time greater than the minimum GCD cycle time, timing windows TW1 and TW2 would, at some viable clock frequency overlap. Of course, (14) is only valid when $n_2 \neq n_1$, but determining whether two timing windows overlap when $n_1 = n_2$ is trivial, and their relative position does not change with T_{gcd}.

The algorithm to find the cycle times when timing windows overlap can be improved by recognizing that a timing window can never overlap a periodic copy of itself, and that sub-LCM windows can be examined for efficiency. For example, if the synchronous group had T, 2T, and 3T clocks, a comparison between a window clocked at T and a window clocked at 2T need only look over a 2T region. Likewise a comparison between a window clocked at T and and window clocked at 3T need only look over a 3T region.

Next we would build a list of GCD cycle times representing when pairs of timing windows overlap. Each member of the list would contain a beginning and ending GCD cycle time and the two windows that overlap in this cycle time range. Two windows that always overlap would be given beginning and ending GCD cycle times of 0 and ∞. The GCD cycle time list would be sorted by beginning GCD cycle times. Finally a series of fixed frequency analyses would be performed.

The first analysis would be performed at the maximum GCD clock frequency using (11) to build the timing windows and the previously-discussed fixed frequency analysis procedure in Section 3.3. To determine worst case noise at the slower clock frequencies, the sorted GCD cycle time list would be traversed (starting at the minimum GCD cycle time) looking for changes in timing window overlap. That is, every new beginning GCD cycle time in the list would indicate that a new pair of timing windows overlapped relative to the preceding noise analysis. And with this change would come the possibility of a yet noisier set of aggressor nets. So at each new beginning GCD cycle time another fixed frequency analysis would be performed, again using (11) to build the timing windows at the new frequency.

3.5 General Analysis

The general analysis would begin by dividing the controlling clocks into synchronous groups. Clocks belong to the same group when they have a fixed and known phase relationship. The C1, C2, C3, and C4 clocks in Fig. 3 belong to the same synchronous clock group because the phase relationship between these clocks is both fixed and known. A source synchronous I/O clock, however, can't be grouped with its source clock because the phase relationship between the clocks, although fixed, is nevertheless unknown.

Next, the timing windows themselves would be identified and grouped into these synchronous clock groups. All timing window edges would be expressed as functions of their synchronous clock group GCD cycle time, T_{gcd}. Note that while the leading and trailing edges of a single timing window can be controlled by different clocks, both controlling clocks will belong to the same synchronous clock group.

Now if any of the nets in the total set of aggressor nets and victim sink is asynchronous to another net in the set, the processing proceeds as follows: If the victim sink is also synchronous to one of these groups, it is included in that group, and the peak noise for the group is analyzed using the previously-discussed process in Section 3.4. Next, the peak noise of every synchronous group that doesn't include the victim net is computed. This computation also uses the variable frequency analysis process, but now we find the peak noise as if the victim sensitivity window extended over the entire LCM cycle. Because the noise from these groups is asynchronous to the victim and asynchronous to the other groups, the noise from each such synchronous group is summed to the synchronously-determined noise (the noise from the aggressor nets in the victim sink's synchronous group) without further pruning to find worst case noise that the victim net can sample. Again accuracy can be enhanced by re-analyzing the coupled network with nonlinear models and simultaneous switching aggressor nets.

4. Other Considerations

Noise affects timing, and vice versa, timing affects noise. An accurate determination of either, even when clock frequencies are fixed, is an iterative process [3]. The procedure presented here addresses one critical aspect of the noise/timing relationship, but it is not an iterative process and it will miss second order effects. While the timing windows can always be padded to prevent these escapes, this added margin will not appeal to every design team. But some design teams, prodded by more demanding manufacturing, test, and system debug requirements, will opt for a methodology that guarantees functionality over a wider frequency range.

5. Conclusions

In this paper we have shown that crosstalk noise is dependent on clock frequency, and have proposed a methodology using timing windows pinned to GCD (Greatest Common Divisor) clock edges that guarantees functionality from dc to f_{max}. Furthermore, we have proposed a more accurate treatment of noise derived from multiple synchronous clocks using LCM (Least Common Multiple) windows for even fixed clock frequency analyses.

References

[1] L.H.Chen and M.Marek-Sadowska, "Aggressor Alignment for Worst-Case Crosstalk Noise", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 5, pp. 612-621, May 2001

[2] R.Levy, D.Blaauw, G.Braca, A.Dasgupta, A.Grinshpon, C.Oh, B.Orshav, S.Sirichotiyakul, and V.Zolotov, "ClariNet: A Noise Analysis Tool for Deep Submicron Design", *Proc. Design Automation Conf.*, June 2000, pp. 233-238.

[3] K.L.Shepard, V.Narayanan, P.C.Elmendorf, and G.Zheng, "Global Harmony: Coupled Noise Analysis for Full-Chip RC Interconnect Analysis", *Proc. Intl. Conf. Computer-Aided Design*, Nov. 1997, pp. 139- 146.

[4] D.L.Rude, "Statisical Method of Noise Estimation in a Synchronous System", *IEEE Trans. on Components, Packaging, and Manufacturing Technology - Part B*, vol. 17 no. 4, pp. 514-519, Nov. 1994.