CROSSTALK DELAY ANALYSIS IN VERY DEEP SUB MICRON VLSI CIRCUITS

Approved by:

_____

Dr. Mitch Thornton, Thesis Advisor, Assoc. Prof. CSE Dept

_____

Dr. John Provence, Adjunct Faculty., EE Dept.

_____

Dr. Sukumaran Nair, Assoc. Prof. CSE Dept.

CROSSTALK DELAY ANALYSIS IN VERY DEEP SUB MICRON VLSI CIRCUITS

A Thesis Presented to the Graduate Faculty of

School of Engineering

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Master of Science

With a

Major in Computer Engineering

by

Satyendra Ravi Prasad Raju Datla

(Bachelor of Engineering, I.E.T.E, New Delhi)

May 15th 2004

ACKNOWLEDGEMENTS

The author wants to take the opportunity to thank all the people that supported this project. In particular, I owe a great deal of gratitude to my advisor Professor Mitch Thornton, who encouraged me all throughout this project and also helped me organize the work. My special thanks to Professor Sukumaran Nair and Professor John Provence for readily agreeing to be on the thesis committee and providing useful suggestions during the project work. I here by thank all the friends and staff at CSE Dept for helping me during the project and providing help during the thesis writing.

I also thank my mother for her unwavering support during the many long days, which went into this endeavor. I am also grateful to all my family members and well-wishers whose love and caring for me always inspired me.

Datla, Satyendra

Bachelor of Engineering,

I.E.T.E., New Delhi, 1999

CROSSTALK DELAY ANALYSIS IN VERY DEEP SUB MICRON VLSI CIRCUITS

Advisor: Professor Mitch Thornton

Master of Science conferred May 15th, 2004

Thesis completed <month>, <date>, 2004

Abstract

Integrated Circuit design has seen revolutionary progress in the past quarter century. Explosive growth of semiconductor applications has happened as a result of several technological breakthroughs. IC design community today embracing sub-100nm wafer design technologies, known as very deep sub-micron (VDSM) technologies, to take advantage of the superior integration possibilities. At these technologies, many phenomena affect gate and wire delays. Capacitive coupling between neighboring wires is one such phenomena that is having significant effect on design's timing and functionality goals. The accurate estimation of these effects is a 'must have' requirement for any design that gets manufactured using the VDSM technologies.

This thesis summarizes the study conducted to identify the root causes of the crosstalk due to capacitive coupling. A case study is conducted on a complex VLSI design to check on the possible effects of the crosstalk on its timing and functionality goals. An efficient analysis and fixing flow is developed and its effectiveness is compared with other available approaches. Some methods are proposed to address the crosstalk problem ahead in the design flow.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Dedicated to my grand mother Mrs. Subbayamma,

Who's continued love, care and support,

I could not be without

1 THESIS OVERVIEW

This report presents the work carried out as part of my master's thesis. The topic chosen for this thesis is one of the current issues the VLSI (very large scale integration) design community is facing today: Signal Integrity. This thesis reports the details of the Signal Integrity problem, the existing solutions, and proposes new solutions to effectively tackle the problem. The thesis has the following sections:

Introduction

The introduction of the thesis problem is given in this section. The background of the issue is elaborated on. Current methods and flow issues are briefly addressed here to prepare audience about the work being carried out.

Crosstalk effects in Very Deep Sub-Micron (VDSM) designs

The crosstalk problem is defined here with more details. The effects of crosstalk are explained with suitable examples and relevant statistical data. The objective of this section is to emphasize the importance of crosstalk analysis in the Very Deep Sub-Micron (VDSM) technologies.

Design details

Details about the design chosen for analyzing the crosstalk are provided briefly here. The architecture and the applications of the design are explained here. This design is proprietary of Texas Instruments Inc. Data is provided to the extent it does not violate any Trademark or Intellectual Property issues. The objective of this section is to provide necessary information about possible issues that might arise due to the nature of the design and its features.

Crosstalk analysis methodology

The crosstalk analysis methodology built using the industry standard EDA tools is explained here. Shortcomings that are possible when using these tools are elaborated. Tradeoffs with different possible methods are also mentioned.

Challenges faced during the analysis

There are many issues that pop-up during the crosstalk analysis phase. Many of these challenges are elaborated with details about their origin and their effect on the crosstalk analysis flow.

Strategies for fixing the violations

Once the crosstalk violations are accurately estimated, there shall be a fix methodology to address these violations. The fix methodology followed to efficiently fix

the crosstalk violations is the main focus of this section. Also, there are some issues that arise due to a particular fix methodology followed. These after-effects are also discussed.

Experimental results

The results of the experiments conducted on the sample design are described in this section. The comparison of regular approach with the proposed approach is addressed.

Lessons learned and few proposals

The focus of this section is to examine the various lessons learned during this research work. Also, some suggestions are made to effectively deal with crosstalk problem at various stages of the design flow. These shall prepare audience with some idea about the challenges one will have to face while performing crosstalk analysis.

Conclusion

Finally, the efficient methodology evolved as a result of the work done is summarized in this section. Future work that can be carried out on this interesting problem is suggested here.

2 INTRODUCTION

Ever since the integrated circuit was invented by Jack Kilby (a Texas Instruments engineer) in 1951, there have been several advances in the technology that led the way the integrated circuits are fabricated. More and more applications have seen silicon taking the advantage of modern integrated circuit manufacturing techniques. The pace at which this integration had been happening is well illustrated by the famous "Moore's law". Per Moore's law, the total number of transistors on a single integrated circuit doubles every 18 months. This law had been very practical from the time the first 4-bit microprocessor was introduced to the current days of Pentium processors. The migration was from few hundreds of transistors on a single chip then to the few millions of transistors on a single chip now. This significant migration was possible only by reducing the feature sizes of the CMOS (Complimentary Metal Oxide Semiconductor) integrated circuit. The feature sizes have moved from few micro-meters to few nano-meters.

These rapid inventions in process technologies allowed integration of very complex electronic circuits into small and tiny chips. As a result of that, very complex features have come to handheld devices. The benefits of integration are only possible after the challenges at each node of integration were overcome. Each of those new nodes certainly brought new challenges, which had to be effectively addressed by the engineering community.

Designing of VLSI integrated circuits have entered new phase namely “Very Deep Sub-Micron” (VDSM) design phase. The designs that are below the feature sizes 0.25µm (micro-meter) fall into this category. As mentioned above, each new design phase has few challenges that need to be addressed in order to effectively utilize its benefits. One of the major challenges that the VDSM technologies face is the signal integrity of the signals on chip. Other possible challenges are well documented in [12]. Signal integrity is the ability of a signal to generate the correct response in a circuit. A signal with good signal integrity has its digital levels at required voltage levels at required times. Because of reduced feature sizes in VDSM technologies, signals in the wires on the integrated circuit suffer with signal integrity issues due to the signal transitions in their neighboring wires. These signal integrity issues have to be kept in control in order for the circuit to function properly.

According to the research conducted by Collett International Research Inc., in the year 2000, one in five chips fail because of the signal integrity issue as illustrated in Figure 1. This is really a significant effect on the yield of the chip production.

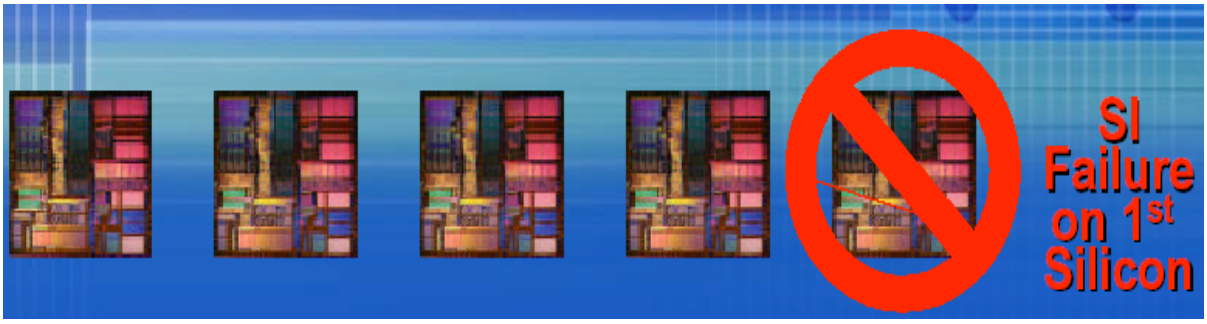

Figure 1. Signal Integrity failures

2.1 What is signal integrity?

Signal integrity describes the environment in which the signals must exist. It covers different design techniques that ensure signals to be undistorted and do not cause problems to themselves or to other components in the system. Signal Integrity did not always matter. In the golden years of digital computing (1970-1990), gates switched so slowly that, on the whole, digital signals actually looked like ones and zeros. Analog modeling of signal propagation was not necessary.

At today's speeds even the simple, passive elements of a high-speed design—the wires, PC boards, connectors, and chip packages— can make up a significant part of the overall signal delay. Even worse, these elements can cause glitches, resets, logic errors, and other problems. So, the signal integrity has been increasingly significant problem in modern VLSI chip designs.

2.2 Signal Integrity issues

Here are the major issues concerning signal integrity:

- Crosstalk Delay
- Crosstalk Noise
- Ringing & Ground bounce
- IR (voltage) drop in power lines
- Electro-migration
- Manufacturing-related issues that if not addressed can lead to chip failure

Crosstalk delay and crosstalk noise are the primary effects of the increased coupling capacitance. The coupling capacitance tends to be more significant in modern process technologies due to small and narrow feature sizes.

All of the above effects are part of the signal integrity. While some of these issues can make designs to fail altogether, some issues force the designs to work at reduced operating frequencies. Obviously, both these effects are unacceptable for system designers who use the chips.

Need for sign-off quality crosstalk analysis and the relevant glossary terms are discussed in more detail [16]. Some of the terms related to crosstalk definition are mentioned here.

2.3 Aggressor versus Victim

Crosstalk is the interaction between signals on two different electrical wires. The one creating crosstalk is called an "aggressor", and the one receiving the effect is called a "victim". Often, a wire can be an aggressor as well as a victim.

2.4 Inductive crosstalk vs. Electro-static crosstalk

The problem of crosstalk can be categorized into two main categories: inductive crosstalk and electro-static crosstalk.

Inductive crosstalk: An electrical current in a loop generates a magnetic field. If this magnetic field is changing, it can either radiate energy by launching radio frequency

waves, or it can couple to adjacent loops ("Inductive cross-talk"). Figure 2. below shows the effect of inductive crosstalk:

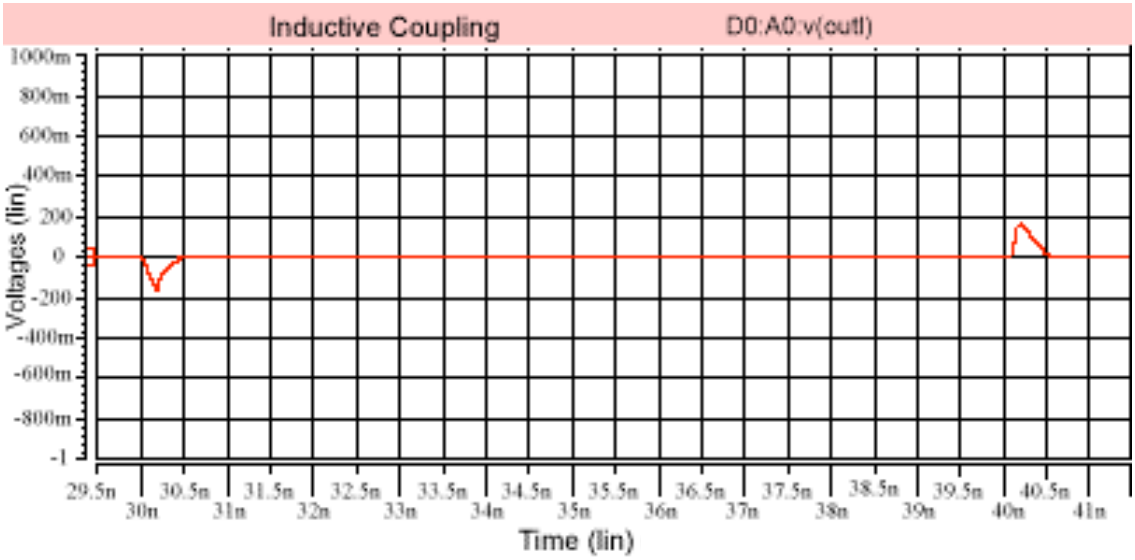

Figure 2. Inductive Coupling

Electro-static crosstalk: The electrical voltage on a line creates an electric field. If this electric field is changing, it radiates radio waves, or it can couple capacitively to adjacent lines ("Electrostatic cross-talk"). Figure 3. depicts the effect of electro-static crosstalk:

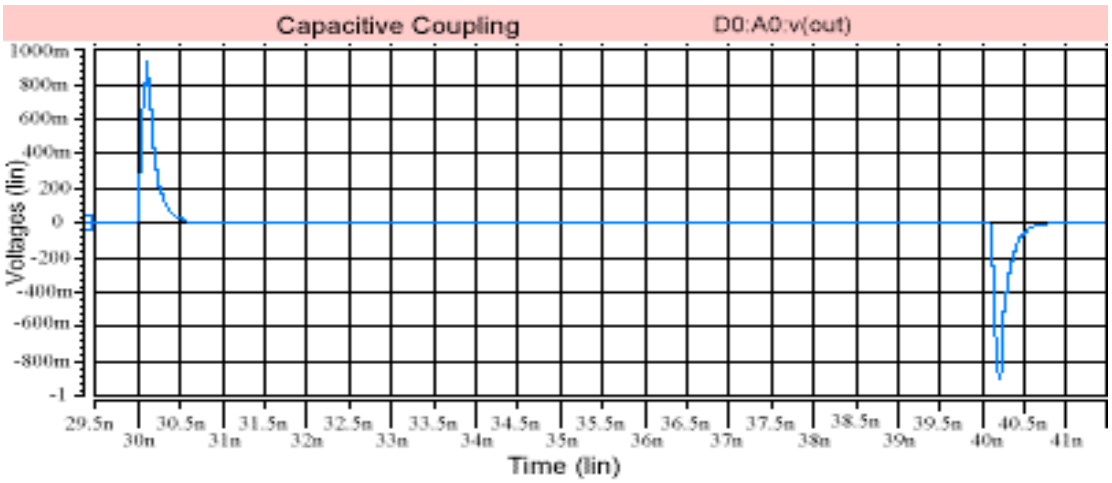

Figure 3. Electro-static crosstalk

As can be seen from the above two explanations, electrostatic crosstalk is significant.

As stated before, the crosstalk problem is a severe one as the geometries shrink beyond 0.25 µm. The following graph effectively illustrates this fact:

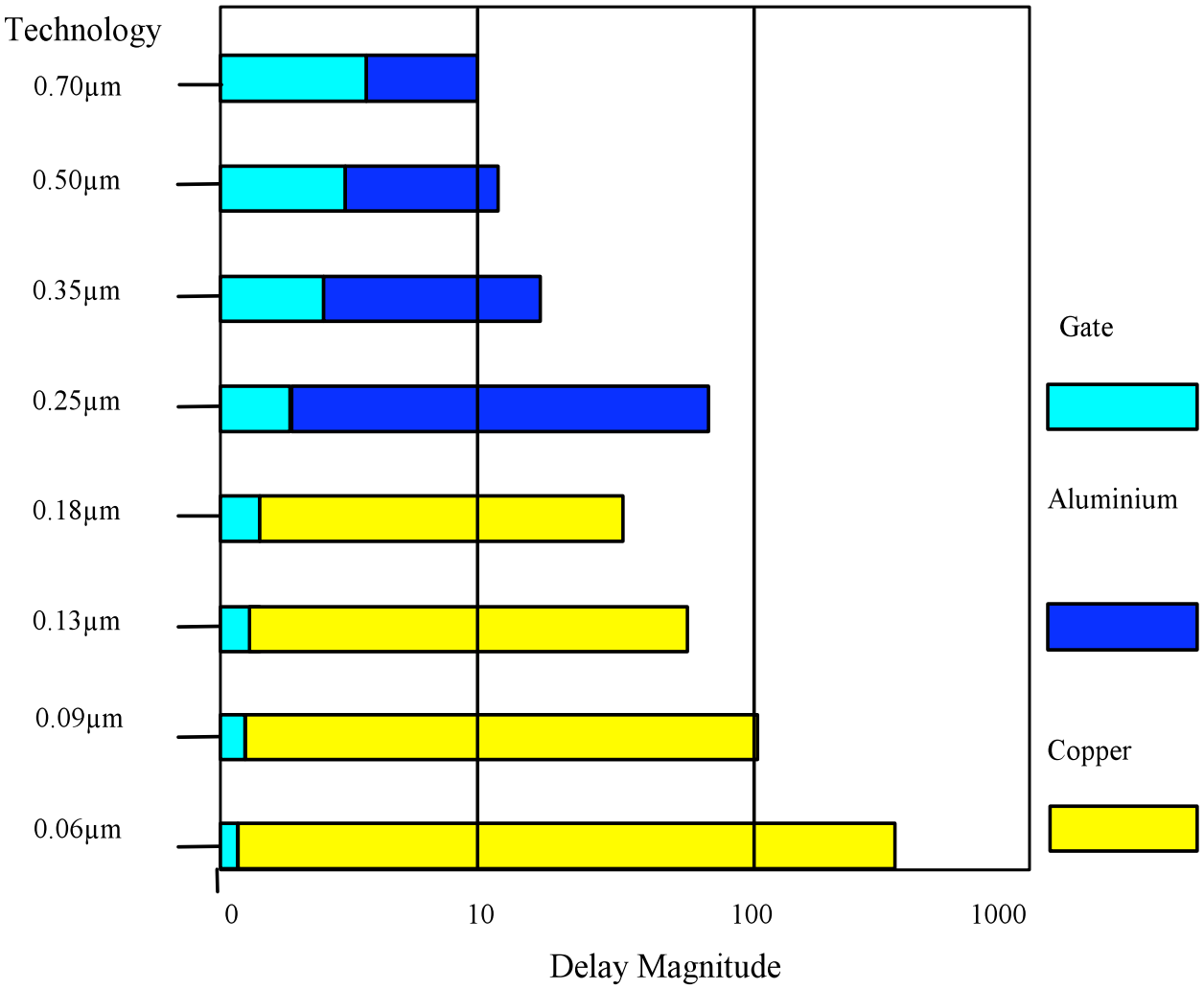

Figure 4. Crosstalk effect vs. technology

As shown in the above Figure 4. at smaller geometries, the gate capacitance is insignificant when compared with wire capacitance. So, the over-all timing path delays are dominated by interconnect delays but not cell delays. The second important point to note from the above figure is that the delay due to coupling capacitance increased almost ten-fold from 0.7 µm technology to 0.09 µm technology. The materials used for interconnect wires were changed from Aluminium to Copper. This brought some positive effect in terms of bringing down the interconnect delay cost. But, at newer

technologies with the reduced geometries, the interconnect delay significantly dominates when compared to the gate delays.

The following Table 1 offers some numbers that support the above concern. Clearly, the interconnect delays did not scale the way the intrinsic delays scaled. More information can be found on this in refs. [20], [23] and [24].

Table 1. Cell Vs. Interconnect delays

Technology (nano meters)	250	180	150	130	100	70
Device intrinsic delay (ps)	70.5	51.1	48.7	45.8	39.2	21.9
1mm wire (ns)	0.059	0.049	0.051	0.044	0.052	0.042

Also, at technologies below 0.09 μm, the standard timing relations become more complex with additional variations and hence bring new issues on to the table.

3 CROSSTALK EFFECTS IN VDSM DESIGNS

As it was shown in the previous section, crosstalk due to coupling capacitance is becoming extremely important in technologies at and below 0.18 micron (Technologies beyond 0.25 micron are usually referred as VDSM technologies). To effectively address the problem of the crosstalk, one must understand the root cause of the issue. In this section, the fundamental causes of the problem are elaborated.

The question that shall be addressed is: "why the coupling capacitance is becoming an issue suddenly?". In modern process technologies, the feature sizes of transistors manufactured are shrinking. The spacing between interconnects is reduced, and hence the coupling capacitance (Cc) is increased proportionally when compared with Substrate capacitance (Cs). Figure 5. below depicts the scenario by comparing the feature sizes at two different technologies.

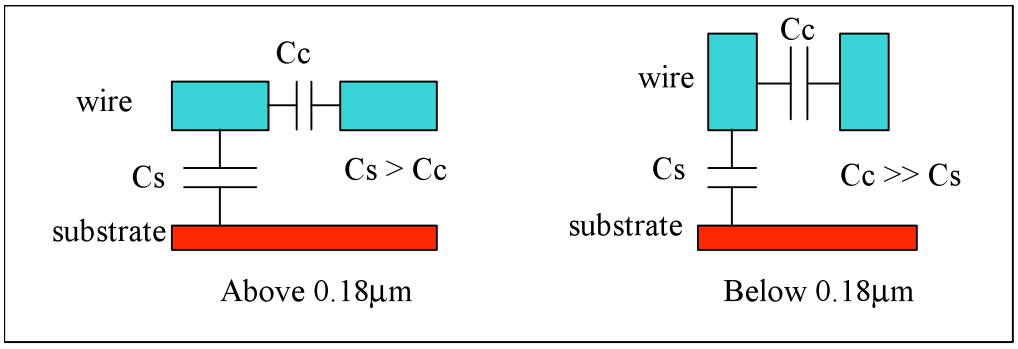

Figure 5. Coupling Capacitance vs. Substrate Capacitance

As shown in the above figure, when the process technologies migrated from 0.18 microns to the smaller technologies, the coupling capacitance values have increased in several orders when compared to the substrate capacitance.

As mentioned in the previous chapter, introduction of copper interconnects helped reduce coupling capacitance (Cc), which still dominates ground capacitance. Reduced transistor switching thresholds have also reduced the noise threshold of logic gates. On the other hand, higher drive strengths are used in latest technologies (helps ease timing closure), which make current designs more noise sensitive.

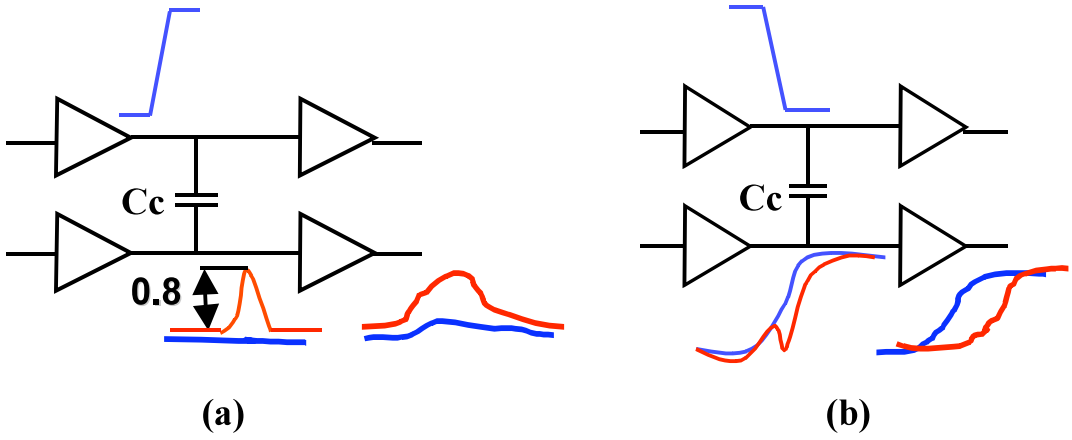

Figure 6. Noise and Delay effects of Crosstalk

Capacitive crosstalk can induce noise (glitches) on a silent (non-switching) interconnect line, and can potentially cause functionality failures (Figure 6(a)). Similarly, crosstalk can cause increased delays when an aggressor switches in the opposite direction of the victim, as shown in Figure 6(b). Conversely, an aggressor can cause decreased delays when switching in the same direction as the victim. This increase or decrease in

delays can cause setup or hold time violations respectively, and may lead to functional failures or reduced operating frequency of the chip. The delay impact due to crosstalk is extremely important, since regular static timing analysis considers all coupled interconnect lines to be quiet, which is seldom the case.

There have been different attempts to accurately estimate the effect of crosstalk on signal delay as explained in [18] and [22]. Since impact of coupling capacitance is very dependent on its magnitude and the drive strength of the adjacent coupled nets, accurate crosstalk analysis can be performed only after detailed routing is completed. Hence, the crosstalk violations show up very late in the design cycle and can cause schedule slips. The current layout tools are not dependably crosstalk aware, making crosstalk prevention and timing closure a challenge.

3.1 Timing effect of Crosstalk Delay Violations

This section deals with various timing issues that are caused by crosstalk. Each issue is described in detail with its cause and effect. The ramifications of crosstalk violations have been well documented in refs. [1], [10], [12] and [13].

3.1.1 Hold violations

Hold violations are possible at sequential elements in the design when the data input does not respect the minimum required hold timing margin. Usually, clock networks are highly susceptible to the crosstalk issue. This is because; they are widely spread across the chip to reach all sequential elements of the design.

In the case of the sample design experiments conducted, the largest effect of crosstalk was the hold timing violations. This primarily happened because one of the clock networks in the design became the victim of a fast switching aggressor. This is depicted in the Figure 7. The clock network has large coupling with another wire that is driven by a large drive strength buffer. The clock network hence becomes the victim of this aggressor as shown.

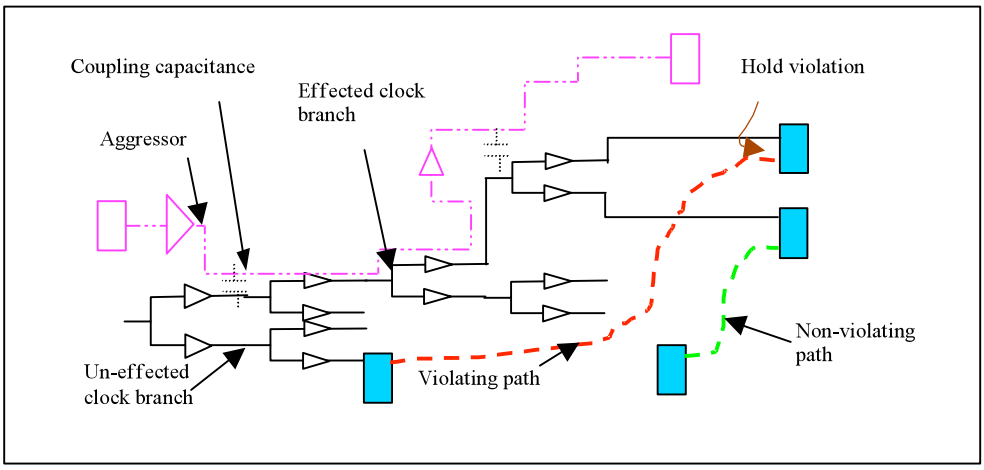

Figure 7. Hold Violations due to Crosstalk Effect

When the aggressor switches in the opposite direction of the clock, clock transitions become little slower. So, the clock transitions reach at the flip-flops little later than they should. Because of this, during hold time analysis some of the timing paths, which use this clock as a capture clock might start failing due to the later arrival of the clock. (Not all paths would show violations, because the same clock might also be used as launch clock.)

3.1.2 Setup violations

Similarly, setup violations are possible at sequential elements when the data inputs do not honor the setup time requirement of the sequential element. Though less number, there are setup violations observed during the crosstalk delay analysis for the experimental design taken. The root cause of these setup violations is explained in the following figure.

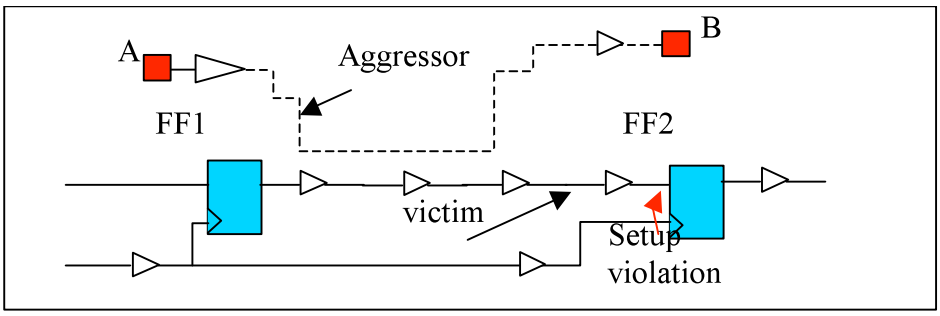

Figure 8. Setup Violations due to Crosstalk Effect

As shown in the Figure 8, a timing path exists between FF1 and FF2. There is a neighboring aggressor path, as shown between A and B. When the aggressor wire is switching in the opposite direction of the signal in the data path (victim), the data input of FF2 could be delayed. Because of this, a timing path that was meeting timing without crosstalk analysis would now show a violation. This is shown in the Figure 9.

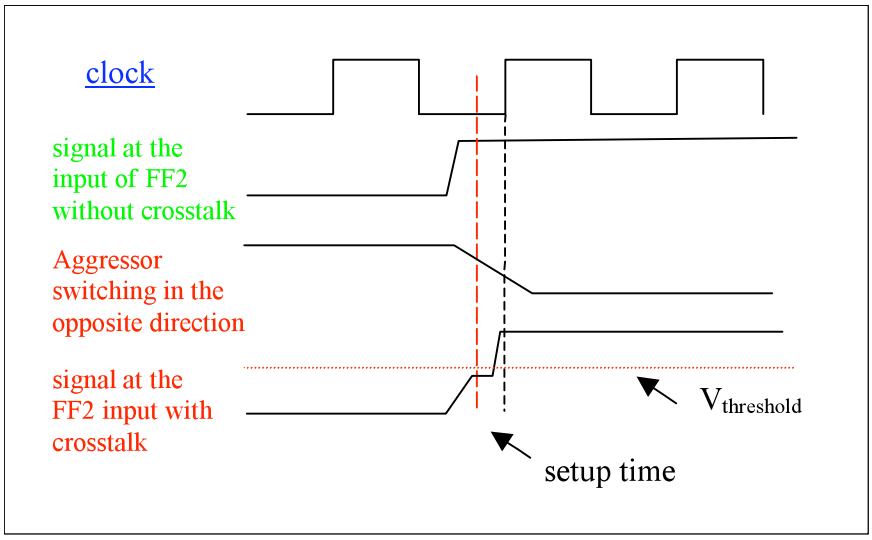

Figure 9. Setup Violation due to Crosstalk Delay

As shown in the figure, originally, the actual arrival time of the signal at the input of FF2 is well ahead of setup requirement of the flip-flop. If the aggressor switches in the opposite direction, the signal is delayed. The flip-flop FF2 now has a setup violation after considering the effect of crosstalk

3.1.3 Bus violations

Bus signals are the other possible victims of crosstalk delay. For example, there are long routed bus signals that connect two far placed blocks. The performance requirements of these bus signals could be very high and hence need frequent repeaters in their path. To reduce the skew among the bits of the bus at the destination, the detailed router tool routes the bus signals together with minimum spacing between them. This is shown in the Figure 10 below:

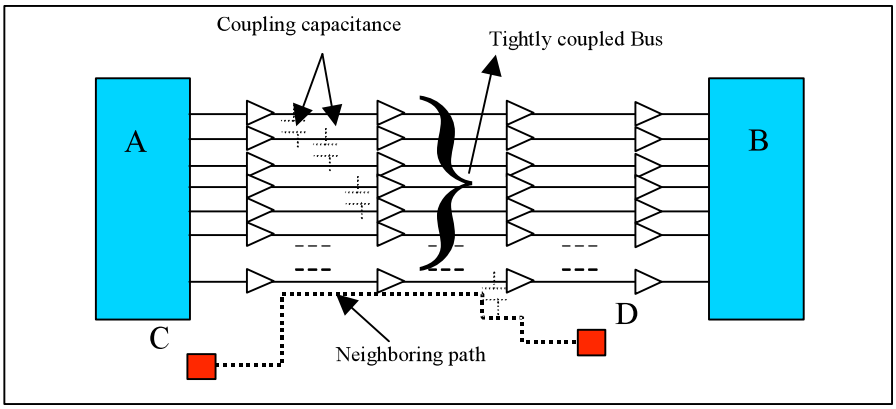

Figure 10. Crosstalk Effect on an On-chip bus

But, usually the bus signals switch together. Due to the simultaneous switching of all the individual bits of the bus, there will be considerable amount of crosstalk that will occur. Hence, the whole bus would behave like noisy transmission line affecting any neighboring wires.

As seen from all the previous issues, crosstalk delay could significantly affect the design. As the problem and its effects are identified, it is now a challenge to address the problem effectively.

4 DESIGN DETAILS

Possible crosstalk violations and their effects on a typical design were elaborated in previous chapter. So, any methodology that is developed shall be verified and proven using a complex VLSI design. The design chosen must be complex in nature and shall represent some of the latest design techniques adopted in the industry. Design chosen therefore can be the best test case for crosstalk analysis and fix methodology.

The design considered for this exercise is a reusable, hardened System on a chip (SoC) core. The design is an ultra-low power, high performance, and open multimedia application platform for 3G wireless applications.

The block diagram, shown in Figure 11, depicts the system architecture of the design. The design is a proprietary and belongs to Texas Instruments Inc.

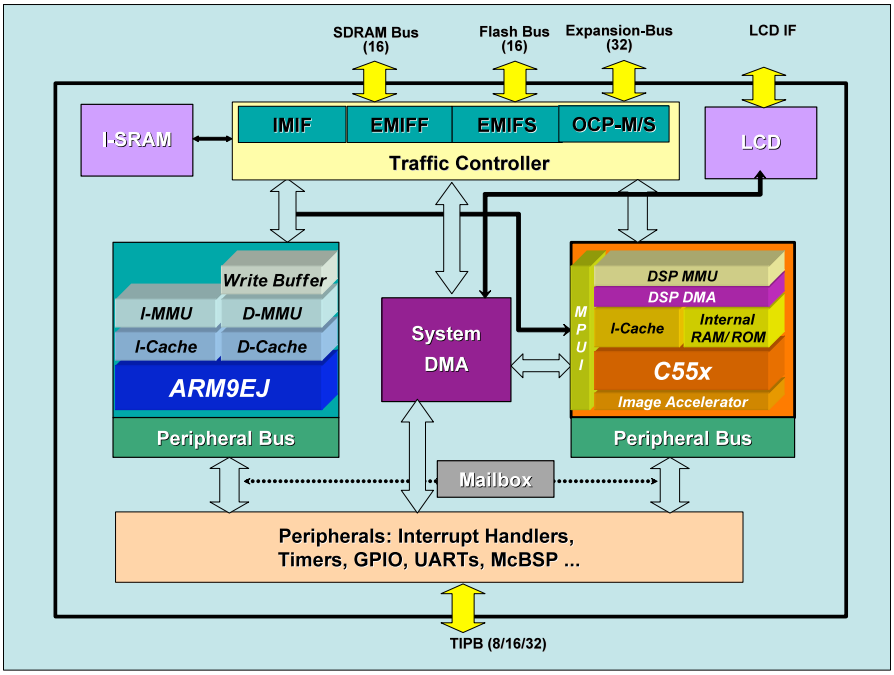

Figure 11. Design Block Diagram

The design integrates a high performance DSP core, based on low power TI C55x™ DSP, and an MPU core based on ARM9™ microprocessor, for the optimal combination of high performance with low power consumption. This architecture offers an attractive solution to both DSP and ARM™ developers. This provides the low power, real-time signal processing capabilities of a DSP, coupled with the command and control functionality of a microprocessor. This platform allows both cores to operate at a speed independent of the system interface, in order to maximize system speed, while at the same time maintain low power consumption. The design is of size approximately 2.5 million gates, including memory blocks. The design is taken as a reference for analyzing the effects of crosstalk delay and the necessary fix identification.

5 CROSSTALK ANALYSIS METHODOLOGY

The main challenge in evaluating the effect of crosstalk on a design is to derive analysis methodology. Such methodology shall be reliable, flexible and yet accurate. More importantly the methodology evolved shall be easily integrated with normal design flow as explained in [13].

So, first it is worth taking a look at the normal design flow. The traditional design flow for a VLSI design is shown in the figure below.

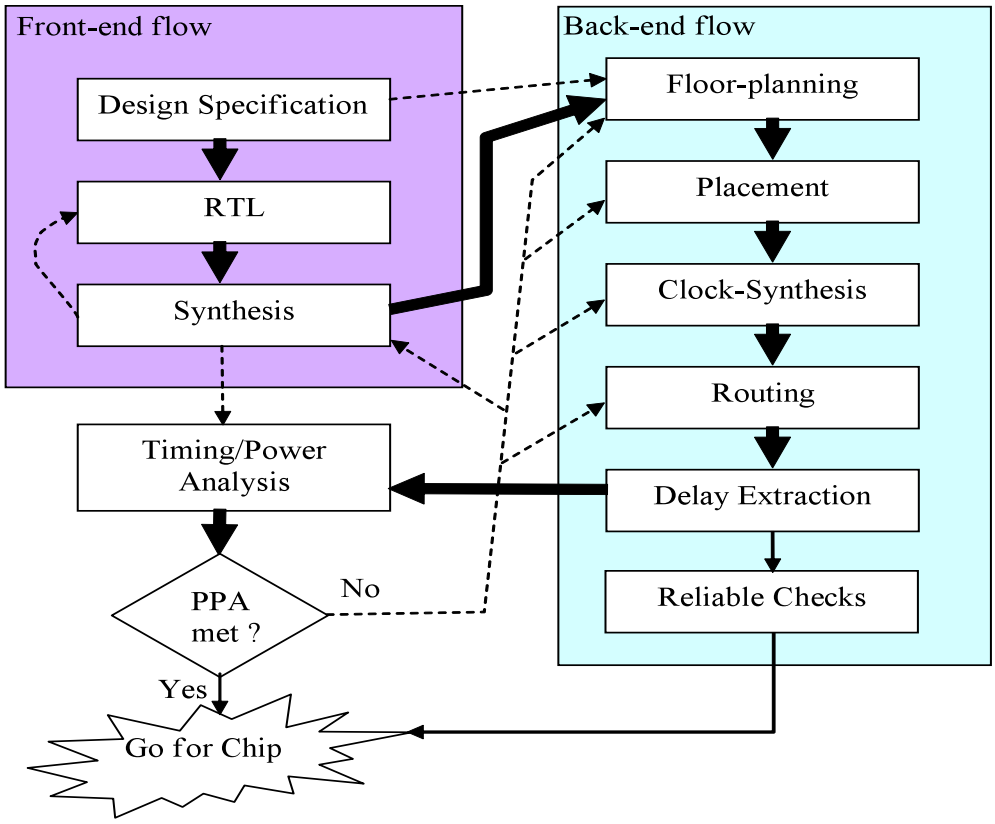

Figure 12. Design flow of VLSI circuits

As shown in the Figure 12, the timing closure loops are performed till the design's predefined PPA (Performance, Power and Area) goals are achieved. As shown in the figure above, the flow can be divided into two main categories. The front-end and the back-end are those two categories. The front-end part of the design flow mainly consists of logic implementation of the given design specification. The design represented in Register Transfer Level (RTL) is synthesized to the target library. The back-end flow starts from the synthesized representation of the design and translates into the physical representation that can then be delivered for chip fabrication. Backend flow involves floor-planning, placement of the standard cells, clock network synthesis, clock routing, signal routing and other reliability check steps. Once the clock and signal routings are completed, the delays can be extracted during the extraction phase.

Crosstalk analysis has to be performed after the detailed routing step in the above-mentioned flow. The detailed routing here means both signal and clock routing. As shown in the flow diagram (Figure 12), the next step is the circuit parasitics extraction. These parasitics are the source for delay calculation.

The circuit parasitics form the basis for any crosstalk estimation as explained in the previous chapters. The following four-step procedure describes the flow developed to perform crosstalk delay analysis:

- Step1: Coupled RC parasitics extraction using EDA tool called STAR-RCXT
- Step2: Generation of the SDFs that include the impact of crosstalk delay incorporated using combination of the delay calculating tool and standard EDA

tool called Primetime. Primetime is an industry standard sign-off tool for static timing analysis.

- Step3: Static Timing Analysis with Primetime, using the crosstalk delay SDFs generated from the above step.
- Step4: Filtering of the crosstalk delay violations based on switching timing windows to identify the violations to fix

Figure 13 describes the flow used for crosstalk delay analysis. The subsequent sections talk about these steps in more detail.

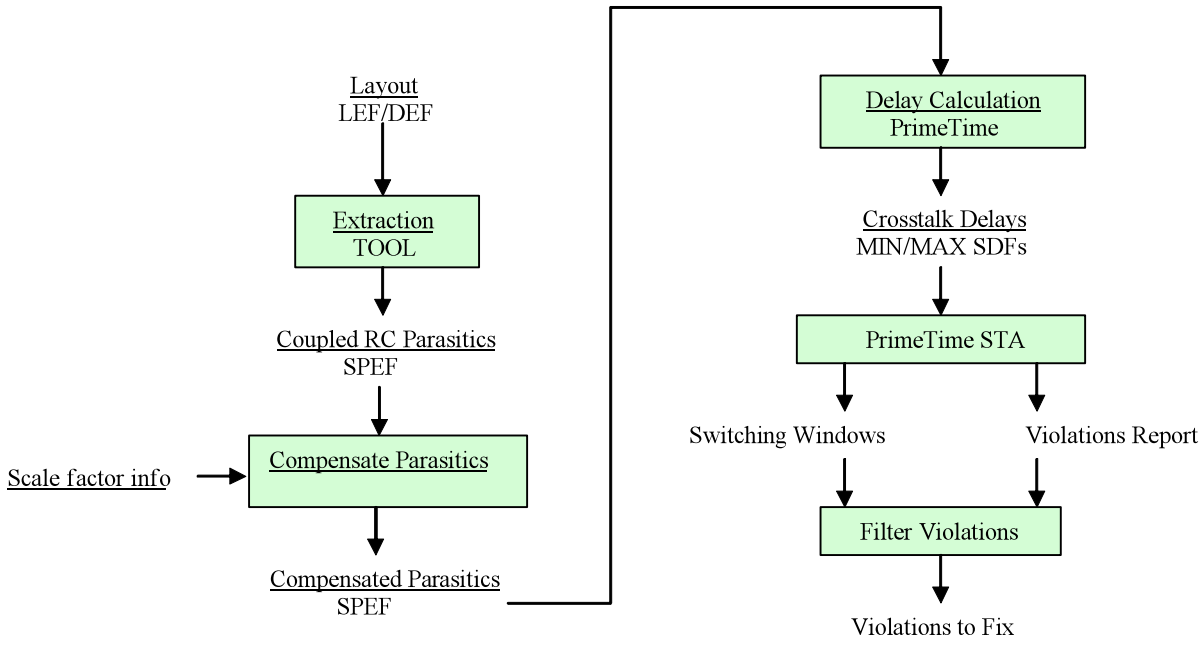

Figure 13. Crosstalk Delay Analysis Flow

5.1 Coupled RC parasitics extraction

LEF/DEF dumped from the layout tool was used as the input to STAR-RCXT for coupled RC parasitics extraction in SPEF format. Main problem with SPEF file is its huge size. STAR-RCXT supports various reduction options to control the size of the coupled SPEF generated without losing on the accuracy.

The following options are used in STAR-RCXT to control the size of SPEF files.

REDUCTION_MAX_DELAY_ERROR, COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD

5.2 Generation of Crosstalk aware parasitics

After generating the net parasitics using STAR-RCXT, a PERL script was used to modify the net parasitics for crosstalk. There are different ways the coupling capacitance impact on delay can be estimated. Three of the approaches are explained in detail in [11]. The methodology used for crosstalk delay computation is an advanced version of coupling compensation method [7]. In coupling compensation, the distributed RC network with coupling capacitances are first converted into an equivalent lumped RC network. Multiplying the coupling capacitance with a dynamic scale factor performs the conversion. The value of the scale factor depends on variables like victim/aggressor drive strengths, victim/aggressor loads and coupling capacitance.

The direction of switching determines whether the delay of the net increases or decreases due to crosstalk (Figure 14). Scale factor (m1, m2) less than 1.0 is used to model same direction switching effect (Figure 14b), and scale factor (n1, n2) greater than 1.0 is used to model opposite direction switching effect (Figure 14c). Since crosstalk causes both speedup and slowdown of paths, it can cause both setup and hold timing problems.

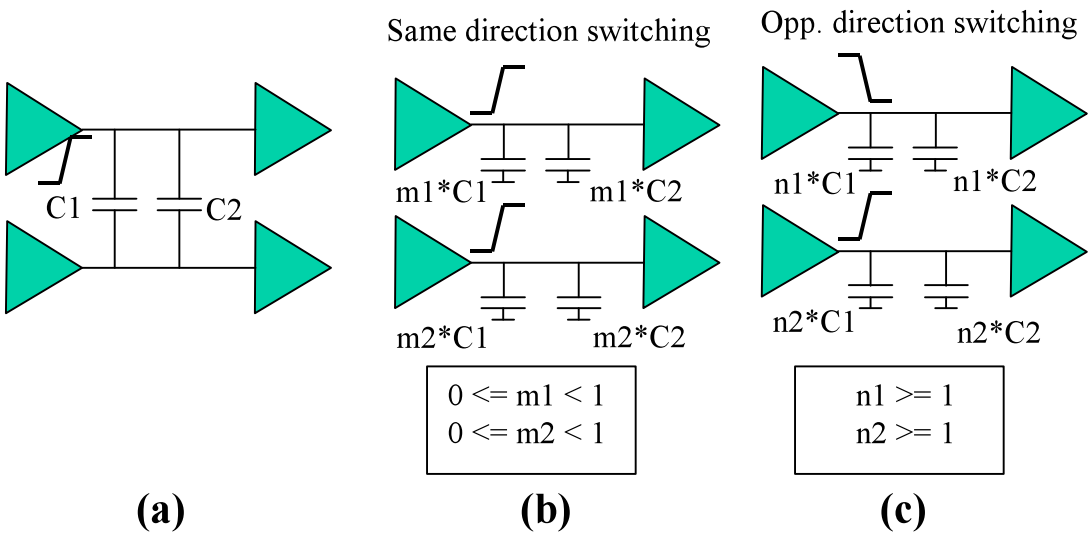

Figure 14. Crosstalk Delay Compensation Approach

All aggressors coupled to a victim are considered when performing coupling compensation. At this point in the flow, all nets coupled to a victim are considered as aggressors. Narrowing the list of aggressors will be performed later (in step 4). This approach makes the crosstalk delay SDF generation step independent of operating modes of the design.

5.3 Generation of Crosstalk SDF

After modifying the parasitics to take crosstalk into account, two-crosstalk delay SDFs are generated for each analysis corner. The following command in Primetime can be used to write SDF:

write_sdf –input_port_nets –output_port_nets –version 3.0 –include {SETUPHOLD RECREM} <SDFfile>

The first SDF file generated is used to model "same" direction switching delays (speedup effect/min switching). The second one is used to model the opposite direction switching delays (slow down effect/max switching).

5.4 Static Timing Analysis using the crosstalk delay SDFs

Once the timing SDFs are generated with PrimeTime as explained in the previous section, static timing analysis (STA) can be performed using PrimeTime. The same timing constraints as the normal timing signoff can be used.

Static timing analysis done with only max or min SDFs cannot reflect a worst-case analysis, as shown in Figure 15.

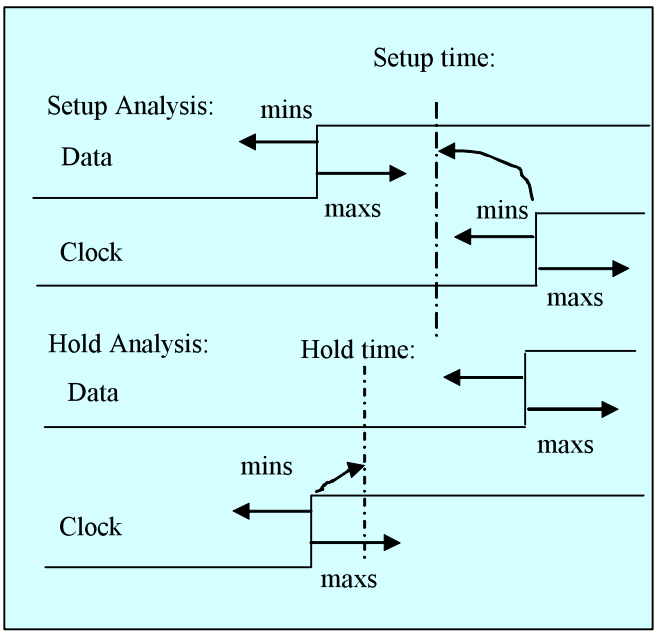

Figure 15. Crosstalk Delay with min and max switching

Annotating the same type of delays (mins or maxs) to *both* clock and data paths makes the clock and data edges shift in the same direction. This can potentially miss real violations.

Because of this, setup timing analysis should be performed by annotating *max* delays to data path and *min* delays to the clock path. Conversely, to perform hold time analysis, max delays need to be annotated to clock path, and min delays to the data path. This can be achieved in PrimeTime by using the on_chip_variation mode of analysis. The following `read_sdf` command is used in PrimeTime to enable the on_chip_variation mode:

```
  read_sdf -analysis_type on_chip_variation \

-min_type <min or max triplet entry> \
```

```
-max_type <min or max triplet entry> \

-min_file <min crosstalk SDF> \

-max_file <max crosstalk SDF>
```

When performing on_chip_variation analysis in PrimeTime, it is advisable that clock reconvergence pessimism be removed. Clock reconvergence pessimism refers to the pessimism introduced in static timing analysis that is caused by using two different delays for the common elements in the clock and data path. By default, Primetime uses different delays for the common elements during on chip variation analysis. The clock reconvergence issue is elaborated with more details in a later section.

With the setup described and the on_chip_variation mode enabled, all the timing violations are reported. The timing violations may not be accurate at this stage. The reason for the inaccuracy is the fact that the coupling delay is applied on the nets with the assumption that all aggressors can switch at the same time. In reality, not all aggressors may switch at the same time. To filter out the incorrect violations, switching timing windows of all the nets in the design are used in the flow.

The switching timing windows for every net in the design are also dumped from PrimeTime after annotating the coupling parasitics. This was accomplished by setting the PrimeTime variable "`timing_save_pin_arrival_and_slack`" to `true`, in order to enable PrimeTime to store the minimum and maximum arrival timing windows for all nets in the design. (By default, PrimeTime stores arrival windows only for the endpoints.)

A TCL script is written to query the pin attribute "`arrival_window`", in order to obtain the arrival windows for all the instance pins in the design. The relationship (synchronous or asynchronous) shall be clearly defined for all the clocks in the design.

5.5 Filtering of violations

The timing path violations obtained in the previous step are then reanalyzed by using the switching timing windows and the clock relationships to identify the violations that really needed to be fixed. Some more discussion on prioritizing the noise violations based on their likelihood of occurrence can be found in [21].

One of the biggest challenges in crosstalk delay analysis is the pessimism in the analysis. Due to the non-availability of dynamic switching activity from simulation, arrival windows from PrimeTime were used to remove pessimism. Here in this flow, many pessimism reduction techniques are used that allowed to narrow down the crosstalk delay violations. Some of the pessimism reduction techniques used are:

- Grouping of synchronous aggressors to a victim, based on aggressor and victim switching windows. This makes it possible to identify the worst-case aggressor set that can cause the maximum crosstalk delay impact.
- Filtering of violations that are caused by inactive clocks when a design is multi-mode constrained. For instance, ignoring violations caused by test mode clocks when in functional mode and vice versa.
- Filtering of violations caused by static nets in the design for the mode of analysis. For instance, signals like scan enable and BIST enable do not switch during functional

mode of operation. So, the violations caused because of these signals can be filtered out from list of functional mode violations.

- Comprehending 1-stage logical correlation between victims and aggressors. For example, if a buffer (or inverter) input and output nets are aggressors to the same victim, it is likely that the timing windows of both aggressors overlap with the victim. But, both the aggressors are logically related (and separated by a buffer delay) and this relation needs to be considered while calculating the cumulative impact of the two aggressors on the victim. A similar explanation is applicable when the same aggressor couples to two victims that are input and output of a buffer (or inverter). Since identifying logical correlation between any two nets in the design is difficult, the restriction applied to the flow is to comprehend logical correlation between two victims/aggressors separated by one stage of logic.

6 CHALLENGES FACED DURING THE ANALYSIS

The flow described in the previous sections is successfully used to conduct crosstalk-aware static timing analysis on the design chosen. While performing the analysis, various issues came up that required special attention. The next sections describe these issues in detail.

6.1 Hierarchical design challenges

In the case of big designs like the one chosen, separate teams usually develop individual blocks. These individual sub-blocks are then integrated. This design methodology can then be called as hierarchical in nature. There are definitely some limitations to perform top-level crosstalk analysis on a hierarchical design. The primary limitation was the lack of data necessary to perform accurate crosstalk analysis on the inter module communication signals. The following Figure 16 depicts this issue.

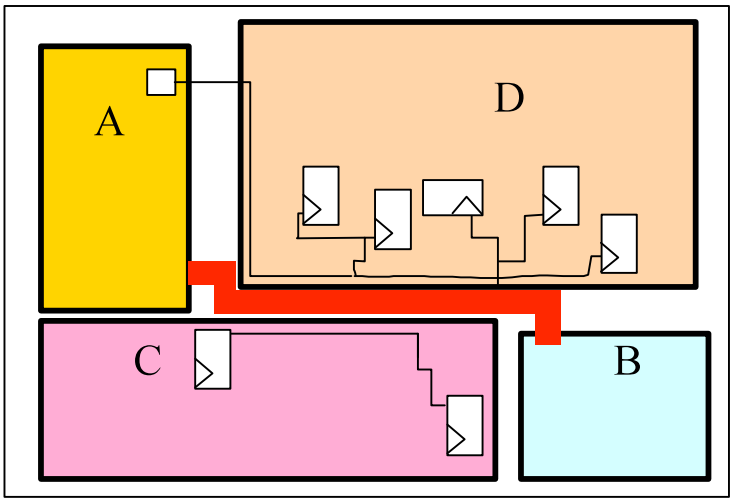

Figure 16. Crosstalk at Block Boundaries

In the figure, blocks A, B, C, and D represent the physical implementation of four hierarchical blocks in a design. There is a bus running from block A to block B, and internal data and clock nets running at the boundaries in blocks C and D.

These internal nets are common at the block level, and hence might not be trouble-makers when performing ***block level*** static timing analysis. At the lower level, the blocks may be meeting the timing budgets without any problems. But, things start breaking when each of these blocks are stitched together and tested with crosstalk-induced delays.

One scenario is when the bus between the A and B blocks switch from its "all 1s" state to "all 0s" state or vice versa. It is interesting to see what can happen. As a result of the excessive crosstalk by this bus, the clock tree in block D might have a longer or shorter insertion delay that usual when there is switching on the bus. This would force many violating timing paths to start popping up inside block D. Similarly, the timing path highlighted inside the block C shall also get affected. Paths that otherwise meet

timing at block level would now start violating at the top level because of crosstalk induced by other signals at the boundary.

So, careful and accurate top-level crosstalk induced timing analysis has to be carried out to determine the root cause of the violations. This is often difficult on multi-million gate designs. While performing top-level timing analysis, for variety of reasons, the team may be using abstract models of the lower level blocks. Extracted timing models like STAMP may represent the module level timing.

Unfortunately, many of these timing models lack the necessary information to perform the accurate top-level crosstalk delay analysis. As indicated above, the essential information needed for top-level analysis is the expanded clock tree of all modules and the timing paths whose nets run at the block boundary. Only then, it is possible to perform accurate crosstalk timing analysis at the top level.

A partial solution to this problem is to use interface logic models (called ILMs). The beauty of ILM timing models is that besides the extracted timing information, all the interface logic associated with the ports can be stored. So, the clock network for all the interface signals can be extracted and available for top-level analysis. This nails down the problem to a manageable size.

The problem is not completely addressed with ILMs either, because ILMs may not contain the timing and physical information of all the paths that run at the boundary of the block. Another disadvantage of ILMs is that some teams may prefer not to have the boundary logic visible for confidential/proprietary reasons. Of course, the most accurate

methodology is to do timing analysis with the entire netlist. But, the analysis with full netlist requires lot of memory and CPU resources.

6.2 Clock Reconvergence Pessimism (CRP) Issues

The design chosen has a fairly complex clock structure in order to achieve ultra-low power goals. The sample clock tree structure is shown in the Figure 17 below:

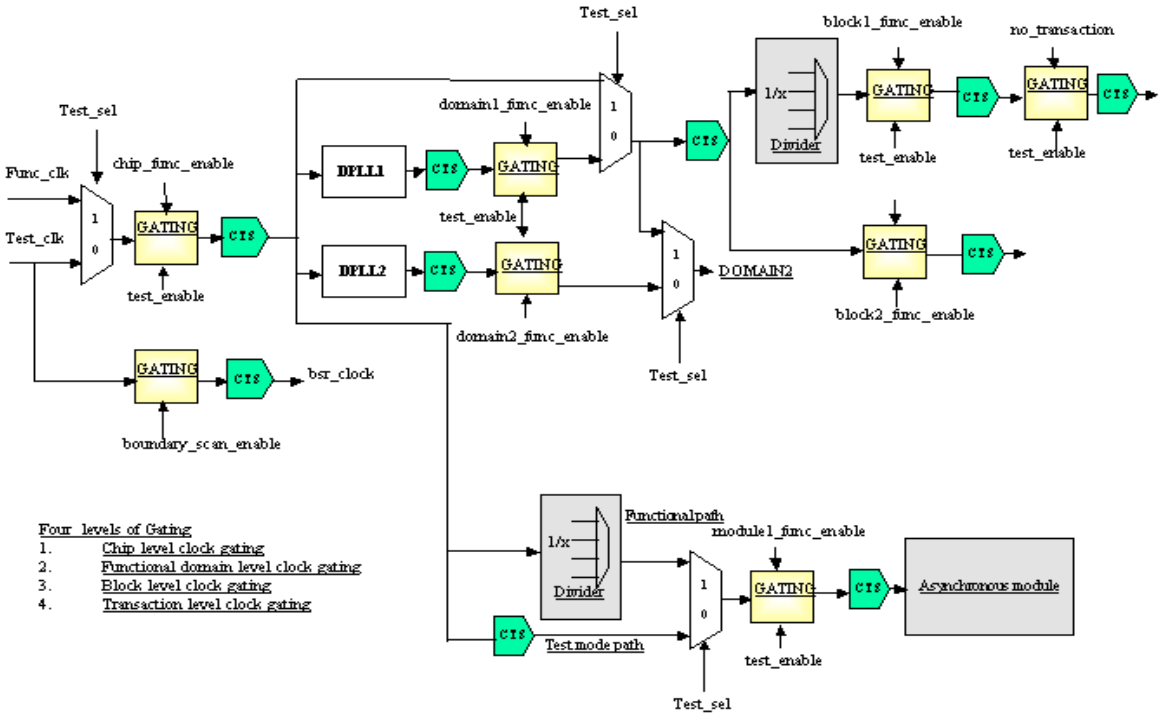

Figure 17. Clock Tree Structure

As shown in the diagram (Figure 17), the clock is gated at several levels to allow for ultra low-power operation by switching off the unused logic/transaction/domain/block. The clock path also has multiplexing, to allow for different operating modes with different clock configurations. Because of this, many of the sub-blocks in the design share derivatives of the common clock from a DPLL (Digital Phase Lock Loop). Even though much of the logic operates on the local clock, there exist some timing paths between the clocks of different branches of the same clock tree.

When performing on-chip variation analysis with PrimeTime, there is some pessimism that needs to be accounted for in order to have accurate analysis. For example, the following diagram (Figure 18) shows a simple two flip flop circuit. The flops share a common clock, but are placed physically at different places on the same die.

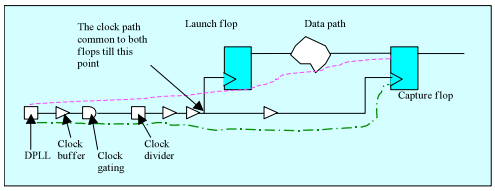

Figure 18. Clock Reconvergence Pessimism

While performing setup timing analysis using on-chip variation mode, PrimeTime uses the annotated max delay for each clock path element of the launching flop, and it uses the min delay for each clock path element of the capture clock. This results in pessimistic analysis, because it is impossible to have two different delays for the same

cell in the clock path for which two delays are used. To remove this pessimism, a variable that needs to be set as shown below:

```
set timing_remove_clock_reconvergence_pessimism true
```

Once this variable is set, PrimeTime removes this pessimism by calculating the amount of pessimism that is induced and adjusts the arrival time by this amount.

With PrimeTime versions prior to 2002.09, there were issues with clock gating macros and other custom clock tree logic. Basically, PrimeTime was not calculating the accurate clock reconvergence pessimism value for paths that contained this logic. Because of this inconsistent behavior by PrimeTime, there are additional violations that are not real.

To address this issue, a PERL script is developed. The script calculates the correct clock reconvergence pessimism value to be removed for the each timing path.

More details on clock reconvergence pessimism and related information can be found in [5].

6.3 Pessimistic Vs. Optimistic analysis

When there exists many number of timing violations, and each one of these violations takes long time to identify and fix. This delays the project execution. The delays are even worse when the fixes require touching some of the paths that already met timing. The primary question becomes: are they true violations? What amount of pessimism is involved?

Crosstalk analysis on a large design is an involved process, and the results vary based on various factors: collapsed and expanded clock trees, aggressor-victim relationships and switching timing windows. Due to lack of firm data, assumptions have to be made for each of these factors. Careful analysis considering these factors can result in reduced violations. The following sections expand on these factors to see the pessimism they cause.

6.4 Collapsed and expanded clock trees

While performing timing analysis, the delay information is annotated by the information extracted from physical design tools. While performing extraction, clock tree information can be extracted by two ways: collapsed or expanded. A collapsed clock tree means that whole clock tree is represented as a single cell. So, the extracted SDF(standard delay format) file contains delay arcs representing maximum and minimum delay possibilities. Collapsed clock trees are usually chosen for reduced run times. Run time becomes an issue when working with larger designs that integrate several sub-blocks.

But, when working with collapsed clock trees, it is observed that the results are more pessimistic. The induced delays due to crosstalk seem to be unfair.

Results are better with less number of violations when the expanded clock tree extraction is used for netlist and delay annotation. This is because of the fact that localized coupling capacitance effects are not seen on a global scale, which is the case with collapsed clock trees.

For example, the expanded clock tree network contains the details of the entire clock tree. As a result, it is possible to see the effect of each sub-net of the clock tree. If a small portion of clock tree were affected by its close proximity with another signal, the delay on clock tree would be reduced or increased. The effect of the additional delay on whole clock tree would vary based on the position of this crosstalk affected partial clock tree.

When using the expanded clock tree, these effects are fairly localized and the accuracy of analysis increases.

6.5 Number of Active Aggressors

It is possible to have multiple aggressors on a single crosstalk victim. The total crosstalk effect on victim is the sum of all the individual crosstalk effects caused by each of the aggressors. But, not all aggressors can be active at the same time and if all the aggressors are considered, then there exists pessimism in the results. Number of violations depends on the number of aggressors chosen to consider at a time on a victim signal. As shown in the Figure 19. below, there is a victim net surrounded by more than one aggressor. When conducting crosstalk delay analysis, there might be a timing violation on the victim, with the assumption that all of the aggressor nets switch at the same time. But in reality, this is not necessarily true. At one time, only few of these aggressors are active, making the initial results pessimistic.

Figure 19. Multiple Aggressors -- Layout Example

The number of aggressors that can be considered to be switching together is a variable that can be changed. For the experimental initial iterations, this number can be chosen as three. The diagram (Figure 19) above shows the case where the victim is routed along with more than two aggressors. The effect on the victim is calculated considering all the aggressors surrounding the victim. This could be pessimistic if the aggressor's timing windows do not match that of victim. So, each violation should be carefully reviewed for the aggressor/victim timing window relations before deciding if it is a violation to be fixed.

6.6 Logically Impossible Timing Windows

The excessive coupling capacitance causes the victim wires to have crosstalk problem. But, coupling capacitance alone is not sufficient for the crosstalk delay effect on a victim. The signal arrival times of both aggressor and victim also need to match in order to have any effect on the timing of the victim. Even if the switching windows of the aggressors and victims match, regular crosstalk delay analysis may still be pessimistic because the current tools do not understand the logic function of the aggressor/victim paths when calculating the timing windows. Two examples are given here to elaborate more on this issue.

In the first case, the Figure 20(a) shows that a1, a2 and a3 aggressors have coupling capacitance on the victim net. Although this coupling capacitance physically exists, it is not possible for all of these aggressors to have an effect on the victim. This is because there is an inverter between a1 and a2.

So, any delay due to same direction switching of a1 would be cancelled by the opposite direction switching of the inverted aggressor a2 assuming both coupling capacitances are identical. Without considering the logical relation information during crosstalk analysis, aggressors a1 and a2 are treated as two different aggressor nets coupling with the victim. That introduces pessimism into the results.

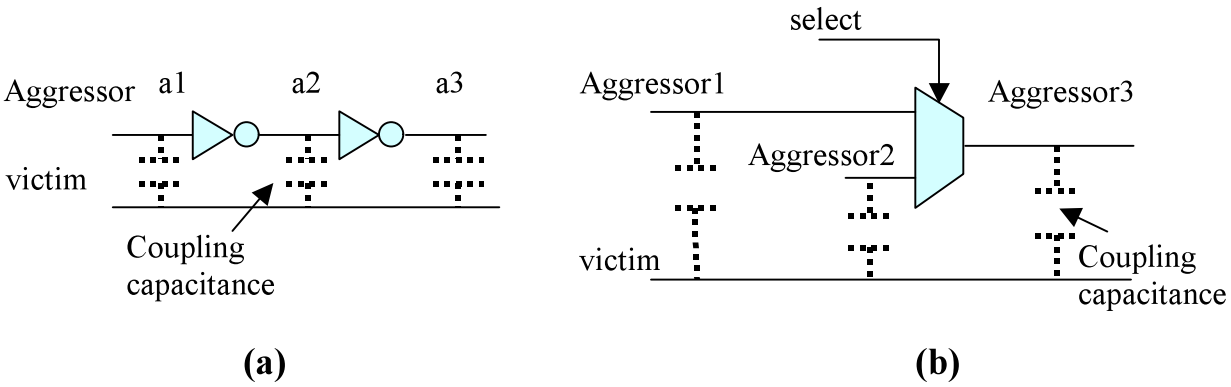

Figure 20. Logically Impossible Timing Windows

The second example is shown in Figure 20(b). Here there is an output of a multiplexer that runs very near to another signal. Physically, the output of this multiplexer may become as an aggressor/victim for/of another net. In this case, output of the mux and both its input nets are aggressors to the victim net. When calculating timing windows of the nets, the tools assume that output of the mux will change whenever one of the inputs changes. This is not necessarily a true situation. If one of the inputs of the mux is selected and the select input is static, then the output of the mux does not have the effect of the timing window on the unselected input.

7 STRATEGY FOR FIXING VIOLATIONS

For the designs of large size and complexity, it is fairly common to see thousands of violations the first time a crosstalk analysis flow is run because of the reasons it was described previously. The challenge is to quickly parse the timing reports generated by the crosstalk delay static timing runs, in order to isolate the real violations for fixing. Unlike regular timing reports, crosstalk-induced timing violations are mainly due to physical routing of the involved nets or paths. So, having access to the physical routing information of the paths is critical in deciding the validity of the violations.

The primary constraint on the methodology for fixing violations is to untouch those paths that have already met timing. There is considerable risk involved when there are any changes to already frozen clock tree structure. So, it is wise to untouch the clock trees unless it was impossible to fix violations without doing so. Therefore, if the violation was because of a clock net becoming the victim/aggressor, then the other net can be re-routed with increased spacing, or repeaters can be inserted to reduce the coupling capacitance effect. This approach paid off well in the experiments conducted where all the violations could be fixed without touching the clock trees. So, no new timing violations are introduced because of crosstalk delay fixing.

Another challenge, of course, is to reduce the number of iterations it might take to fix the violations. Hold fixes especially could require several iterations if not carefully

analyzed and fixed. Methodology followed is to fix all the setup violations that are less in number and then concentrate on fixing the hold. Because hold fixing can be done easily by introducing some buffers.

The following flow diagram (Figure 21) shows the steps followed to fix the crosstalk violations:

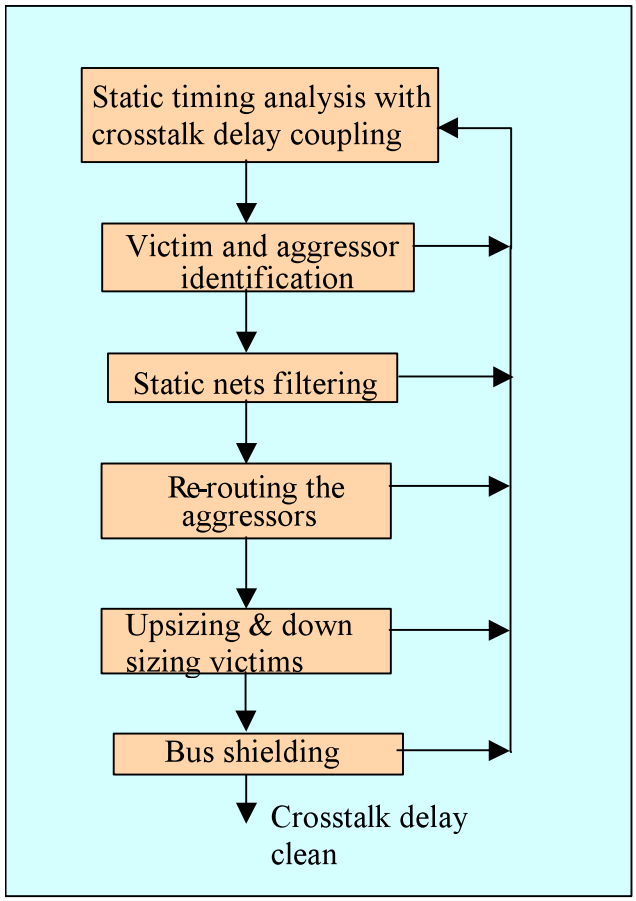

Figure 21. Flow Used for Fixing Violations

As shown in the Figure 21, crosstalk violations fixing is a highly iterative process. There are several iterations for identifying and fixing the violations. Each of these

iterations includes one or more steps as shown in the Figure 21. Each of the steps in the flow is detailed in the subsequent sections.

7.1 Identification of Aggressors and Victims

It is impossible to manually check the thousands of violations reported due to initial crosstalk delay analysis. A PERL script is used to parse the PrimeTime generated timing reports that contain all the crosstalk delay information. The script parses all the timing reports to identify aggressor and victim information for each violating timing path. The other inputs for the script are the coupling capacitance information of all the nets in the design. The script then identifies various aggressors and victims based on the amount of delay effect caused by net coupling.

Aggressors and victims are then sorted based on the clock domain they belong to. Usually, each path group contains a few aggressors and victims that are responsible for many of the violations. So, the result of the PERL script was parsed again to get a list of unique aggressors and their victim nets. This brings down the number of nets to be analyzed to a reasonable number.

7.2 Filtering of static nets

Once the actual aggressors and victims are identified using the above step, the next step is to filter the static nets out of this list. Static nets are the nets whose value does not change in the current operating mode. For example, the “test mode” signal value never

changes during the functional mode of operation. List of all the static nets is prepared for each operation or timing mode. Other static net examples include reset signals and boot-up configuration registers.

The idea is that these static nets could safely be removed from the list of aggressors, since they do not really switch during normal operation, so there is no question of crosstalk with other nets. As a result, the false crosstalk violations are filtered in the process by declaring the static nets.

7.3 Clock network isolation

As mentioned earlier, the primary concern is to minimize, if not avoid, changes to the existing clock tree while fixing the crosstalk violations. This is because, the design's timing is sensitive for any minor changes in the clock trees and may have to go through multiple iterations to close timing. So, the clock networks are not touched during the crosstalk delay fixing.

Therefore, clock network nets are isolated from the list of aggressors and victims by carefully reviewing the timing reports. A full clock timing report from PrimeTime is generated using the following TCL command within pt_shell :

"*report_timing –from <launch> -to <capture> -path_type full_clock_expanded"*

This feature helps to get the fully expanded clock network path for the violating paths. These reports help to identify the clock networks easily.

Basic rule of thumb to be followed: if aggressor is a clock net, then victim is marked for fixing. On the other hand, if victim is a clock, then the aggressor shall be marked for fixing. If the clock network is both an aggressor and victim, still the clock networks shall not be touched. Then, the total crosstalk on the victim can still be addressed by fixing the next level of aggressor/victims.

7.4 Re-routing the Aggressors and Victims

Once the list of actual aggressors and victims are identified with the help of the above steps, the appropriate ones shall be fixed in order to resolve the crosstalk violations. The primary fixing approach shall be to attack the problem with minimal impact on the design. So, it is nice to see if the routing tool could re-route some of the aggressor nets incrementally using increased spacing rules. The other constraint is not to touch the other nets while doing this incremental routing. The routers usually do a better job of handling only a small number of nets incrementally.

The re-routing is done using the Apollo place-and-route tool on the experimental design. The actual procedure followed along with an example script can be found in [1].

This step usually reduces the number of violations from several hundreds to few tens. After performing various other scenarios on the experimental design, it is identified that fixing the violations by additional routing space is the best step to try first in order to make life easier for sub-sequent iterations.

7.5 Up-sizing/down-sizing

Another approach that can be followed is to up-size the cells (ex: buffers, inverters) that drive the victim nets in order to allow victim nets to have enough drive strength to reduce the effect of coupling from aggressor nets. A similar approach is to downsize cells that drive the aggressor paths in order to reduce their effect on the victim nets.

This method helps as a secondary alternative. But, there are situations where this approach may not be usable. For example, there is a scenario in which the aggressor and the victim are mutually coupled and each of them is aggressor/victim to each other. So, any up-sizing/down-sizing might solve the issue in one direction but would worsen the issue in other direction. The approach that can be followed to fix such violation is a combination of spacing and breaking the nets with repeater insertion.

7.6 Splitting the Aggressors and Victims

The coupling capacitance caused by the aggressor is proportional to the length of the net. The longer the net, the more the coupling capacitance it contributes.

The long interconnect nets causing crosstalk violations can be identified and can be carefully broken into multiple nets by inserting repeaters. Only constraint that shall be considered is the physical location of the new repeater being inserted. The criteria that can be used in choosing a particular place to insert repeater is to make sure that the new repeater does not create any new aggressors/victims on other neighboring paths.

7.7 Bus shielding

The long interconnect bus signals between blocks can be shielded by Ground (VSS) wires on both sides of the bus. This shielding avoids the potential excessive crosstalk otherwise possible. The shielding hence reduces the bus violations.

This approach was proved as very effective fix in the tests run on the experimental design.

8 RESULTS OF THE EXPERIMENTS

Once the flow is established, it is required to measure the quality of results (QoR) the flow is yielding when compared to other available methods or flows. The number of iterations taken for timing closure and fixing the crosstalk violations could be the gauging factor when comparing the different flows.

This section summarizes the results of the fixing methodology developed during the period of this research work. The steps mentioned in the previous chapters were followed during each and every iteration of the crosstalk analysis and fixing flow. This methodology yielded pretty good results in terms of successfully closing the timing with a fewer number of iterations. The following graph (Figure 22) gives the iterations and the respective approximate violation numbers.

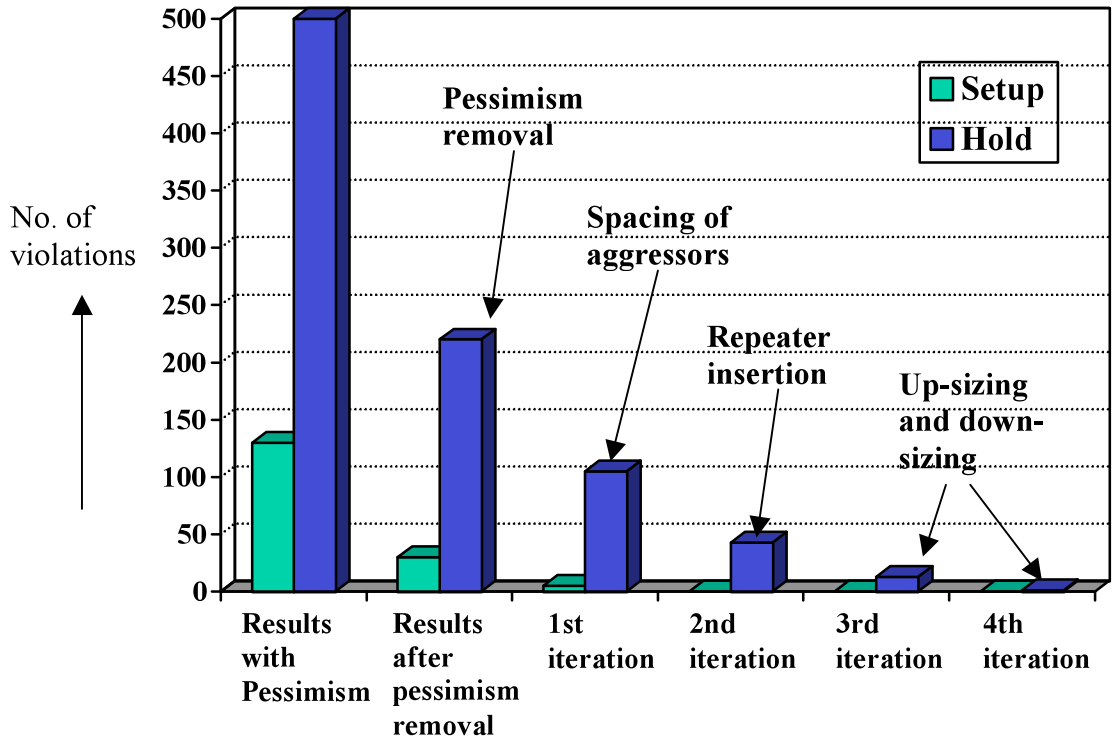

Figure 22. Crosstalk Delay Analysis Fix – Iterations

As seen in the figure, the hold violations are the dominant violations in the design. The initial violations seen before any post-processing are in thousands. These are caused mainly due to too much pessimism in terms of multiple aggressors and asynchronous clock group assumptions as explained in previous sections. The timing for crosstalk delay is closed after 4-5 iterations. Each iteration mentioned here means from the step the crosstalk delay STA is done to the step of fixing the violations and handing-off back-annotation to STA.

The Quality of Results of the developed flow/procedure is compared with traditional flow/procedure to measure the competence of the flow being proposed. For a complex design, two flows are compared with each other in terms of number of timing closure

iterations each flow taken for closing the design for all crosstalk violations. The comparison results are shown in the following Table 2.

Table 2. Crosstalk violations comparison

Flow	Violations	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Iter 6	Iter 7	Iter 8	Iter 9	Iter 10	Iter 11	Iter 12
Traditional Flow	Setup	150	120	100	130	150	60	75	40	12	20	3	0
	Hold	400	500	520	380	220	300	50	80	20	35	8	0
Proposed Flow	Setup	150	80	30	5	15	1	0	0	0	0	0	0
	Hold	400	300	45	70	20	5	0	0	0	0	0	0

The pictorial representation of these results is given below. The main comparison is between the traditional flow and the flow that is developed during the period of research work.

Figure 23. below shows the results of traditional timing closure flow for crosstalk as explained in [19]. The setup and hold violations are sensitive to each other. Most of the times, fixing setup violations would make the number of hold violations increase and vice versa.

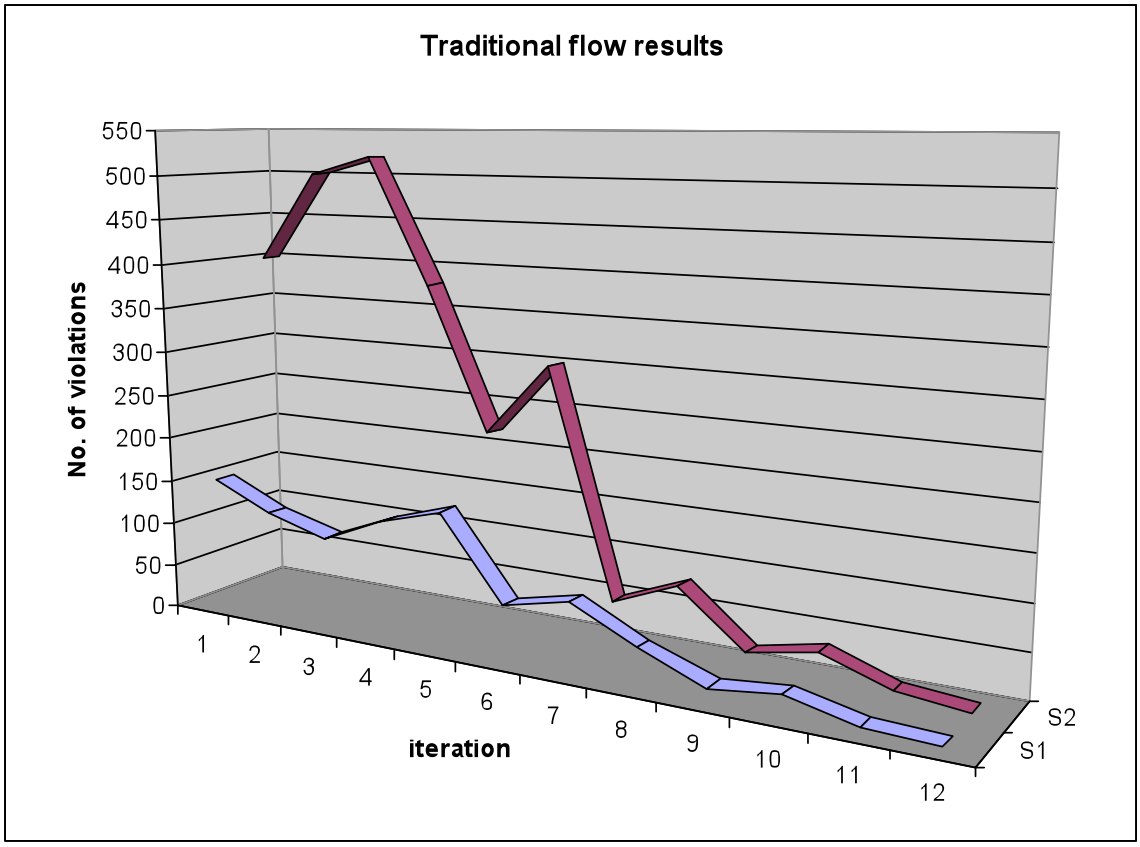

Figure 23. Traditional Flow results

The results are plotted as shown in the Figure 23 above. Clearly, this traditional flow requires 12 timing closure iterations. The reason for these long iterations is also apparent from the above picture. When we try fixing the setup violations, the number of hold violations increases and vice versa. The problem is complicated because of the fact that the effect of any fix being done is not noticed until after parasitic extraction. New fix might bring totally different scenario in terms of coupling capacitance. So, this complicates the fixing methodology and design teams tend to spend more time than afforded in fixing the crosstalk issues. The result is that the total timing closure cycle increases and hence the development cost of the chips.

As discussed in the previous section, this problem can be solved by systematically addressing the root causes of the problem. This fact is proven by conducting experiments through derived methodologies for comparison. The Figure 24 depicts the timing closure results of these trials.

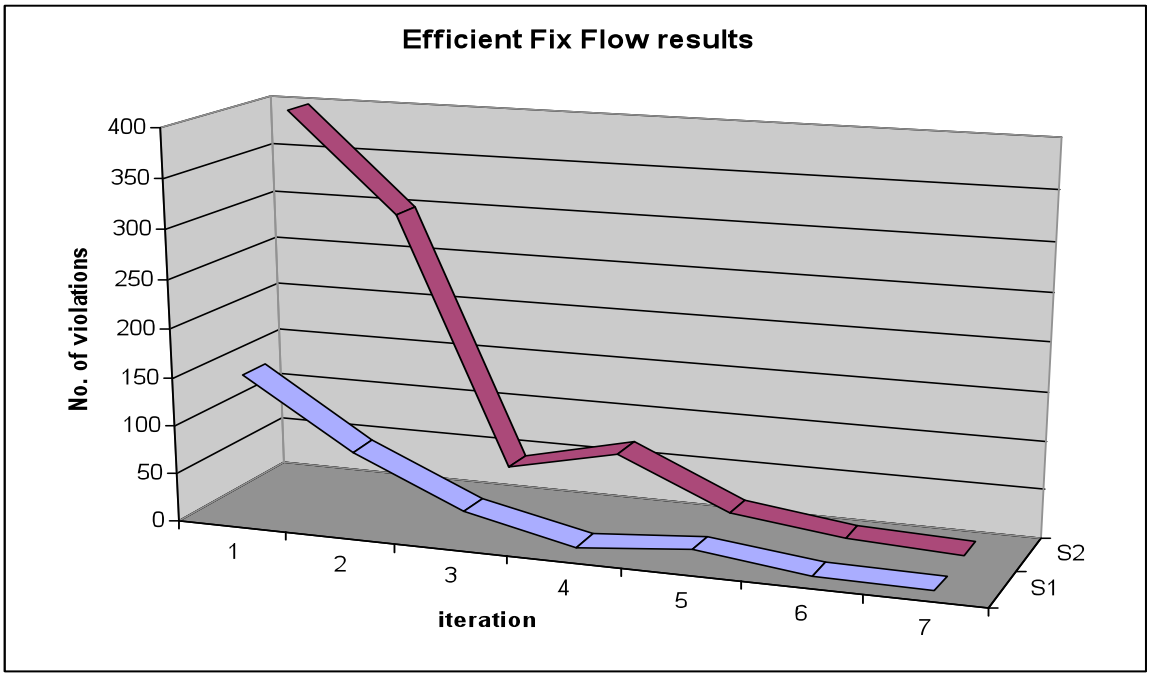

Figure 24. Efficient Fix methodology

The design is taken to go through the timing closure loops with emphasized focus on crosstalk aware methods at each of the design flow steps.

The obvious result is the reduction in number of iterations taken for timing closure. The reduction of number of iterations is definitely a phenomenal advantage to have in this highly competitive VLSI design arena. The time-to-market is the key for majority, if not all, of the modern applications.

9 LESSONS LEARNED AND PROPOSED GUIDELINES

During the course of this research work, the root causes for the crosstalk problem are studied. Also, the possible solutions to efficiently fix the problem are studied and verified the concepts on experimental design. As shown in the previous section, the proposed methodology is very efficient and results oriented.

At the same time, there are still some places where things would have been much better placed if some of the precautionary actions were taken ahead in the flow. As a result of that, several lessons are learned during the course of this work. These lessons learnt may be useful to the VLSI design community to address the crosstalk issue well ahead in their design flow to avoid some last-minute surprises.

Also, there are few novel approaches that can be used to address the crosstalk issue from the beginning of the flow. These new ideas and guidelines along with the lessons learnt are detailed in this section. Each of these lessons and proposals are grouped together into relevant part of the whole design flow.

9.1 Attack the issue from the beginning

Address the issue from the beginning of the design flow. The best way to address a problem is try to avoid it! The design and timing closure flow shall be built with crosstalk aware methodology in view from the beginning.

9.2 Specifications phase

The specification phase of the design is the earliest stage where the initial planning for the design is done. The logic partitioning usually happens depending on the features that need to be supported by the design. One of the after effects of improper logic sharing is the excessive crosstalk due to long wires running between the different parts of the design. So, the logic distribution across various hierarchies shall be crosstalk aware.

The long running bus wires can be avoided by carefully distributing the logic where it is mostly used. If necessary, logic shall be duplicated at the places where it is mostly needed instead of running the long interconnects to long distances across the chip.

9.3 Micro-architecture phase

Some of the crosstalk issues faced late in the design cycle can well be avoided with better implementation of the design during the micro-architecture phase. In order to implement the required functionality, there are many choices during the micro-architecture design phase. But the selection of improper implementation could make things worse for crosstalk. At the same time, there are few implementations that are preferable compared to other from crosstalk perspective. The following examples provide an insight into how this is possible.

Example #1: A large counter is the requirement in a design. If it is implemented as a regular binary counter, then there will be lot of crosstalk that can result when the counter

values change. Also along with the size of the counter, the risk of crosstalk increases. This can be avoided by choosing the counter implementation as Gray counter instead of a binary counter. Because in Gray counter, only one of the bits changes at a time, it reduces lot of crosstalk.

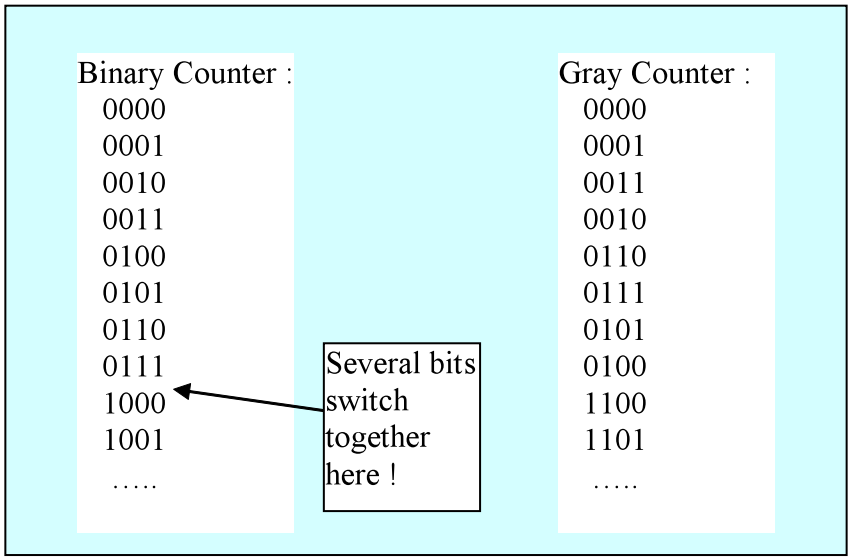

Figure 25. Binary Vs. Gray

As shown in the Figure 25 above, the binary counter has many places of multiple bits switching together. Whereas in gray counter only one bit switches at a time.

So, the selection criteria for counter implementation shall be crosstalk aware.

Example #2:

Large state machines can be avoided. Finite State machines (called as FSMs) have become very popular among the design community as they offer solutions to several implementation challenges. There are different flavors of these FSMs depending on the way outputs are generated and states are incremented. Along with the advantages, FSMs

come with few additional drawbacks. One of the most important drawbacks is that the complexity of the FSMs increases with their size and number of inputs they have to keep track with. Larger state machines make it compulsory to have large counters and the related combinational circuits. This drawback can be avoided by breaking long state machines into smaller and more modular state machines. The recommendation here is to limit the size of the state machine to 20 states maximum.

9.4 Logic Synthesis phase

Logic synthesis is the phase in which the micro-architecture implementation is mapped onto target technology library. The technology library contains the variety of standard cells with basic functionalities like AND, OR, NAND, NOR, INV and DFF etc. These standard cells are pre-designed for the given technology. Logic synthesis tools would translate the implementation from Hardware Description Languages (HDLs) into netlist of those standard cells from the library.

The proposal here is to control the slew rate through constraining the design during the synthesis. Proper slew rate control from the beginning would avoid bigger aggressors later in the design cycle.

The following Figure 26 depicts the constraint for the typical 90nm technology:

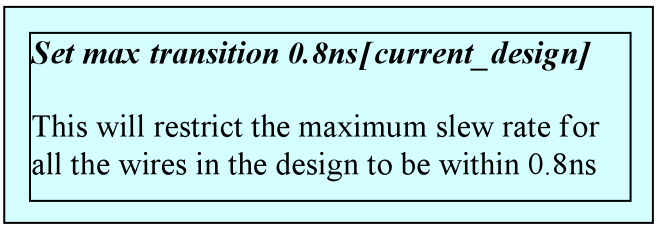

Figure 26. Slew constraining

9.5 Floor-planning phase

Floor planning is the first crucial phase in the physical design portion of the design. The Floor-planning phase consists of tasks like die size estimation, port placement, macro placement and power routing etc. There are several opportunities at this phase to try and avoid as much crosstalk as possible. These following examples would be detailing on things we can do:

Example #1: Proper port placement

Placement of the ports is very critical with respect to the issue of crosstalk. Especially, the ports that have more probability of switching at the same time shall not be placed next to each other. This worst possible switching can be avoided by choosing proper port placement.

For example, in a typical microprocessor based system, large address and data bus signals are the part of the ports. It is customary to group all the address ports together during the port placement. This will have significant effect of crosstalk both inside and outside the chip for these ports. Possibility of all the address and data bits switching

together is large and can cause potential timing issues due to heavy crosstalk. The effect could also be extended on all the neighboring wires if their switching windows overlap with switching windows of address and or data bus.

Proposed solution to this problem is to interleave the address and data bus signals. The reason for that is, the possibility of simultaneous switching of address and data bus bits is less. This will reduce the potential crosstalk problem otherwise possible. The Figure 27 below depicts the scenario:

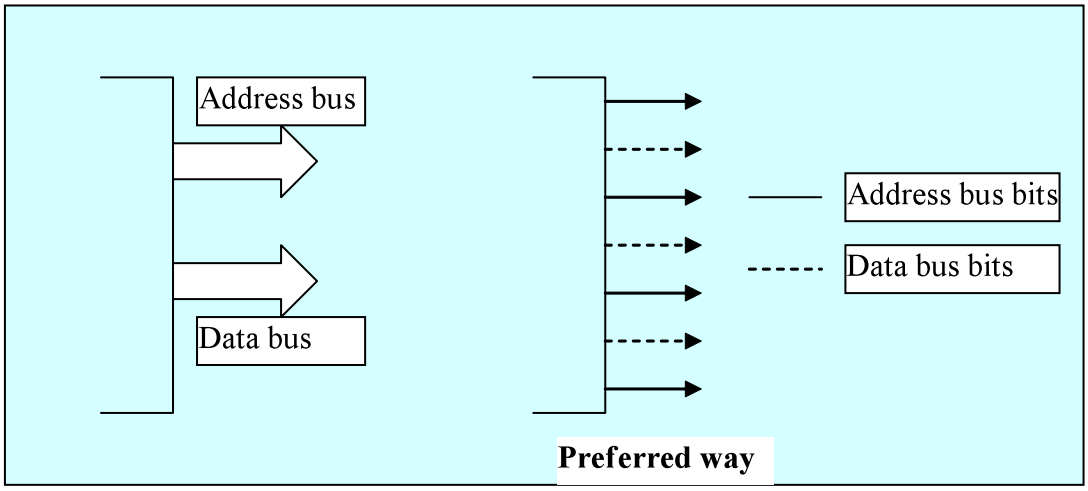

Figure 27. Bus interleaving

Example #2: Macro placement

Placement of the macros inside the chip is critical from the crosstalk perspective. The macros like DPLL and memories need to be placed at the places where they are mostly connected. The signals from these macros may need to be traveling to long distances from different places of the chip. Especially if the design has special high frequency

macros whose signals need to be routed to long distances, they may create possible crosstalk with neighboring wires.

Example #3: Shielding for bus signals

Even with special care taken, there will be cases where bus signals may have to run long distances. So, this issue can be avoided by shielding the bus signals with Ground wires. The shielding will be helpful in reducing the effect of crosstalk by grounding the excessive coupling from the bus lines.

The effects of number of ports and the packaging on crosstalk are explained in more detail in [17].

9.6 Placement phase

Placement is the phase in which the standard cells are placed within the cell area that is prepared during the floor-planning phase. Currently, the EDA tools available are mature enough for better placement results. With the advances in computing resources, the placement tools are doing considerably good job in achieving better timing closure. Lately, many of these Placement tools also implement the crosstalk avoidance algorithms.

However, the case is different from design to design. Additionally, due to new physical effects in each new technology node, placement tools may need to be properly used to get optimum results. From crosstalk point of interest, placement tools shall be properly constrained in order to achieve less crosstalk delay problems.

For example, placement blockages can be created at the areas where there is high switching activity possible. This way some of the possibilities of crosstalk can be avoided.

9.7 Clock distribution phase

Clock distribution and synthesis are the phases in which the clock networks of the design are constructed for the clock signals in the design. Increasing complexity of clock networks comes from the fact that more and more power saving features are implemented in the design.

As the performance requirements of the designs keep increasing, clock frequencies are growing up and up. Due to reduced spacing between the wires in the latest technological nodes, clock networks with heavy switching are highly susceptible to crosstalk. Special care taken during the clock distribution phase can be helpful in avoiding the possibility of harmful crosstalk later in the design cycle.

For example, the transition time of the clock network shall be kept well under control during the clock synthesis phase. Higher the transition times on the clock, more the probability of having crosstalk with its neighbors.

Secondly, the types of the clock buffers chosen during the clock synthesis also would be critical for the smooth timing closure during the Engineering Change Order (known as ECO) phase where the minimum changes are essential. Usually, it is preferable not to use the set of highest and lowest drive strength buffers during the initial clock synthesis. Later on during the final timing closure iterations, these higher/lower driver strength

buffers can be used by upsizing/downsizing the existing buffers. If the higher drive clock buffers are chosen in the beginning of the synthesis phase, there will be little choice towards later stages of timing closure.

Another thing that can help to reduce the crosstalk is to have special routing rules for all the high frequency clock networks. For example, the clock networks could be routed with constraints like double spacing (2 times the technology allowed spacing between the metal wires) and double VIA (Whenever clock signal switches from metal to metal, there will be double VIA placed instead of single VIA). At sub-100nm technology nodes this constraint is becoming a requirement.

9.8 Routing phase

During the routing phase, connections to all the placed instances (as explained in the Placement section) are established and routed with available metal resources. Routing tools shall achieve the basic job of establishing wire routing while honoring the spacing and geometry rules defined by the technology node. The increased need for routing tools to avoid crosstalk problems is well addressed in [15].

The important thing to take care in avoiding the crosstalk during the routing phase is to guide the routing tools with the design information in hand. Following are examples, which are on this track :

- Creation of special routing guides for the high frequency switching nets.

- Special routing constraints across the narrow channels, if any.
- Conservative spacing rule constraints for high frequency switching nets. For example, double spacing constraint for clock nets etc.

The necessity of minimum channel routing is explained in [4].

9.9 Static Timing Analysis phase

The Static timing analysis(referred as STA) forms a very important timing closure step during which the design's timing is verified with predefined timing goals. The STA is usually performed at various stages of the design flow: after initial synthesis, pre-route phase and post-route phase etc. Where as the Crosstalk STA can only be performed after the routing is completed.

The recommendation here is to have additional margins in closing the regular timing. These additional margins would help in accounting for the possible delay degradation because of crosstalk.

10 CONCLUSION

This section concludes the work presented so far in previous chapters.

Signal integrity is one of the critical problems the VLSI design community is facing today. Crosstalk delay is the major contributor for signal integrity issues at latest process technology nodes. The actual basics of the problem are explained in the beginning of this report. The effects of these crosstalk violations with the deep sub-micron design phase are explained in detail with examples.

Once the basic understanding of the issue is established, the details about a complex design that is taken for conducting the experiments are explained. Potential areas where the crosstalk would affect this sort of design are discussed.

The most important focus of this research work is to come up with methodology and procedures to address the problem. The flow developed during the course of this research work is elaborated in the subsequent sections.

The two basic ways of addressing the crosstalk issue are proposed:

- Avoiding the possibilities of the crosstalk
- Fixing the existing crosstalk issues.

Both these methods are discussed in detail in this report. Evaluation of proposed methods is conducted and the results are presented. Lessons learned during the course of this project are summarized in this report.

All the objectives aimed for are achieved at the end of the project. However, there is future work to be continued more on this direction. Following are the potential areas where future work is needed:

- Design tool methodologies to measure the effect of signal integrity ahead in the design flow to help system and chip level designers.
- Improving accuracy in the results obtained from the flow.
- Potential reduction of run times for several of the crosstalk timing closure flow
- Automation: Complete automation of analysis and fix methodology flow proposed in this thesis

11 REFERENCES

1. Satyendra R. P. Raju Datla, “Crosstalk Delay Threat: Are you ready?”, SNUG, Boston 2003. (This presentation received 4th place among the conference papers)

2. James Song, Stewart Shankel, Satyendra R. Datla, Kaijian Shi and Yuanqiao Zheng, “Challenges in the Hierarchical STA of a Low-Power 3G Wireless Application Platform”, SNUG, San Jose, 2003

3. Satyendra R. Datla, James Song, Stewart Shankel, Kaijian Shi and Yuanqiao zheng, “Overcoming challenges in STA for 3G wireless application”, Comms Design Journal, May 2003, EE Times Journal, June 2003.

4. T. Gao and C.L.Liu, “Minimum crosstalk channel routing”, in Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, 1993.

5. Ken Wong and Elisabeth Moseley, “Static timing analysis with crosstalk delay with Primetime SI”, SNUG 2002.

6. Sonke Grimpen, “Delay calculation with detailed parasitics in Primetime”, SNUG 2001.

7. A. B. Kahng, S. Muddu and E. Sarto, “On Switch Factor Based Analysis of Coupled RC Interconnects”, DAC 2000

8. Kei Hirose and Hiroto Yasuura, “A bus delay reduction technique considering crosstalk”, DATE (Europe) 2000.

9. Synopsys Solv-Net CRPR_AppNote_v1.0.pdf

(http://solvnet.synopsys.com/retrieve/customer/application_notes/attached_files/005511/CRPR_AppNote_v1.0.pdf?1049952177065)

10. Bret Victor and Kurt Keutzer, "Bus Encoding to prevent crosstalk delay", ICCAD 2001.

11. Ravishankar Arunachalam, Karthik Rajagopal and Lawrence T. Pileggi, "TACO: Timing Analysis with Coupling", DAC 2000.

12. Dennis Sylvester and Kurt Keutzer, "Getting to the Bottom of Deep Submicron II : A Global Wiring Paradigm", SNUG Europe 2001.

13. Bijan Kiani and Anthony Hill, "Static crosstalk analysis assures silicon success", EETimes Journal, June 2002.

14. Franzini, B., Forzan, C., Pandini, D., Scandolara, .P and Dal Fabbro, A. "Crosstalk Aware Static Timing Analysis Environment", ISQED 2000.

15. Tak Young, "IC Layout must avoid crosstalk problems", EEdesign Journal, June 2002.

16. George Mekhtarian, "How to Achieve Sign-off Quality Signal Integrity Analysis with Prime Time SI", Synopsys Compiler magazine, May 2003.

17. Francesc Moll and Miquel Roca, "Effect of package parasitics and crosstalk on signal delay" (www.spi.uni-hannover.de/2001/presentations/moll.pdf)

18. Sachin S. Sapatnekar, “Capturing the Effect of Crosstalk on Delay”, VLSI design conference, 2000.

19. Dennis Sylvester, Chenming Hu, O. Sam Nakagawa and Soo-Young Oh, “Interconnect Scaling : Signal Integrity and Performance in Future High-Speed CMOS Designs”, Proc. Of I, ZSI Symposium on Technology, pp 42-3, 1998.

20. Sarma B. K. Vrudhula, David Blaauw and Supamas Sirichotiyakul, “Estimation of the Likelihood of Capacitive Coupling Noise”, DAC 2002.

21. Martin Kuhlmann, Sachin S. Sapatnekar and Keshab K. Parthi, “Efficient Crosstalk Estimation”, IEEE ICCAD Conf., 1999.

22. Jason Cong, “An Interconnect-Centric Design Flow for Nanometer Technologies”, Proceedings of the IEEE, 89(4):505-528, April 2001.

23. Jason Cong, “Challenges and Opportunities for Design Innovations in Nanometer Technologies”, SRC Design Sciences Concept Paper, NC, 1997.

24. P.D. Gross, R. Arunachalam, K. Rajagopal, and L.T.Pileggi, “Determination of worst-case aggressor alignment for delay calculation”, in Proceedings of the IEEE/ACM International Coference on Computer-Aided Design, November 1998.

25. O. S. Nakagawa, et al, “Closed form modeling of on-chip crosstalk noise for deep submicron ULSI interconnect”, Hewlett Packard Journal, August 1998.

26. D. A. Kirkpatrick and A.L. Sangiovanni-Vincentelli, “Techniques for crosstalk avoidance in the physical design of high-performance digital systems”, in Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, 1994.