2004

Behavioral and RT-level estimation and optimization of crosstalk in VLSI ASICs

Suvodeep Gupta
University of South Florida

Behavioral and RT-Level Estimation and Optimization of Crosstalk in VLSI ASICs

by

Suvodeep Gupta

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Srinivas Katkoori, Ph.D.
Nagarajan Ranganathan, Ph.D.
Hao Zheng, Ph.D.
Wilfrido Moreno, Ph.D.
A. N. V. Rao, Ph.D.

Date of Approval:
November 1, 2004

Keywords: word-level statistics, probability, floorplanner, routing, synthesis

DEDICATION

To my family and all others who inspired me to persevere through trying times.

ACKNOWLEDGEMENTS

I would like to thank Dr. Katkoori immensely for his guidance, encouragement, and patience during the entire course of this research. I would like to thank Dr. Ranganathan, Dr. Zheng, Dr. Moreno, and Dr. Rao for being on my committee. I take this opportunity to acknowledge the intellectual and moral support provided by current and past members of the VCAPP group, especially Chandramouli, Stelian, Hao, Saraju, and Ananth. I would like to acknowledge the help provided by the Computer Science Tech Support team led by Daniel Prieto. Most importantly, I would like to thank my family for making me believe in myself besides much more. Finally, a big thank you to my friends for being there for me throughout this roller-coaster ride.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

BEHAVIORAL AND RT-LEVEL ESTIMATION AND OPTIMIZATION OF CROSSTALK IN VLSI ASICS

Suvodeep Gupta

ABSTRACT

Downscaling of technology causes signal integrity problems due to crosstalk between closely-spaced interconnect lines. Existing crosstalk estimation and optimization techniques operate at the layout-level of circuits and fail to utilize the efficient design-space exploration at the high-level. To address this, we propose word-level statistical techniques which estimate crosstalk between bus lines: (1) Given a data stream, the first technique simply counts the number of crosstalk events on each bus line. The drawback of this technique is that the execution time is proportional to the stream length. This is overcome by the second *enumerative* technique which is purely statistical in nature. (2) Given word-level statistics, we estimate the bit-level crosstalk probability of bus lines. (3) We further speedup the statistical method using a *non-enumerative* technique by linearizing its complexity with respect to the bus width. Average errors of less than 15% are obtained for bus-widths ranging from 8b to 32b while execution times are reduced by two orders of magnitude, compared to HSPICE.

We then measure the crosstalk susceptibility of nets in the post global routing phase (performed using CADENCE Silicon Ensemble), prior to detailed routing using (1) P_t, the probability of crosstalk on victims in different regions along their route; and (2) V_{peak}, the maximum crosstalk noise amplitude experienced by victims along their route. P_t is estimated using the fast and accurate statistical estimator we previously proposed. V_{peak} is estimated by predicting the cross-coupling capacitances between neighboring wires, using their global routing information. Average errors are less than 8%, compared to HSPICE.

We combine the crosstalk susceptibility values from individual regions along a victim wire's route, to obtain a single susceptibility value for the entire wire.

Further, we propose a register binding technique during high-level synthesis to minimize crosstalk at the register outputs in the RT-level design. It involves modification of the clique-partitioning algorithm to make crosstalk-aware choices of edges to be mapped to the same register. RT-level comparisons between the regular and crosstalk-aware designs show upto 16% reduction in crosstalk activity at the register outputs.

CHAPTER 1

INTRODUCTION

The last four decades have seen the realm of VLSI design expand at a breathtaking pace. Starting with the *microprocessor revolution* in the sixties, the digital circuit domain has witnessed unprecedented changes in both technology and design style. The circuit components have shrunk in size while the total number of components within a fixed chip area has increased manifold, in accordance with the empirically-predicted Moore's law. The combined effect of these factors has resulted in scaling down of technology into the very-deep-submicron (VDSM) and ultra-deep submicron (UDSM) regimes where the physical distance between different components has become extremely small.

Existing technology enhancements have increased circuit speed and complexity while simultaneously increasing power consumption and shrinking area. However, at the VDSM and UDSM technology regimes, designers are now faced with new challenges. The interaction between the circuit components has become significantly different compared to previous technologies and has given rise to new effects such as coupling, IR drop, electromigration,and leakage power dissipation. These effects, which were inconsequential before, have become critical design parameters with immense effects on the functionality and reliability of the circuit. As a result, techniques to estimate and optimize these effects have become increasingly important research areas. It is predicted that these effects will not only alter the conventional design flow but will even change the existing device structures in the foreseeable future [3].

Further, the world-wide demand for smaller and more portable devices has been increasing at a rapid pace. This is corroborated by the rising sales of PDAs and palmtops in the commercial market. In order to keep up with the rising demands of the market, new and

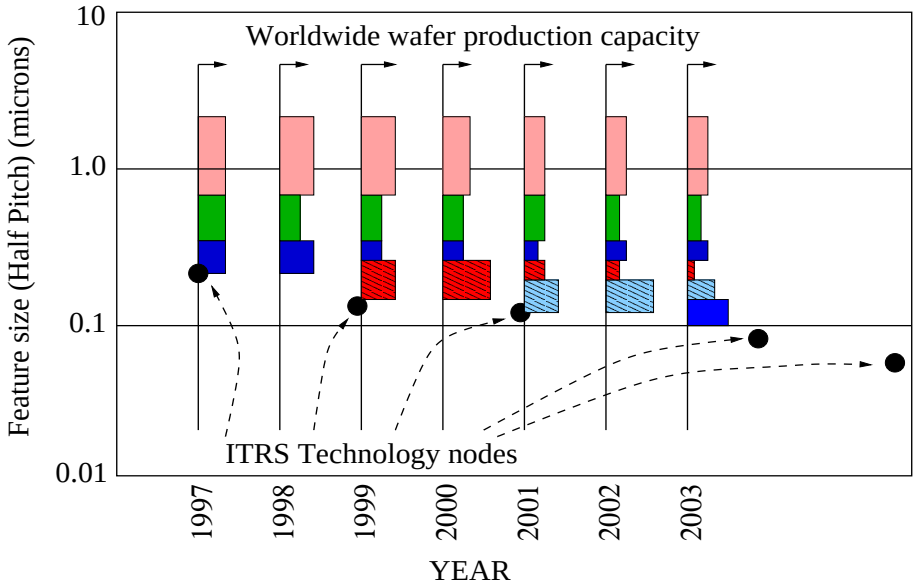


Figure 1.1. Scaling down of technology nodes (ITRS 2003)

improved techniques for extensive miniaturization of circuits is called for. Figure 1.1 indicates the scaling down of technology nodes in circuits as well as the worldwide production of circuits belonging to various feature sizes, as projected by the International Technology Roadmap for Semiconductors (ITRS) 2003 [3]. The figure shows that the technology size has been constantly decreasing over the years. Besides, the world-wide production of end-products with larger target technologies is also diminishing. Miniaturization of circuits causes reliability to become an issue of concern. On one hand, noise sensitivity of these circuits should be kept at a minimum in order to preserve reliability. At the same time, dynamic power consumption in these devices must be kept at a minimum in order to enhance their lifetimes. We analyze the effects of changes in the supply voltage on these two design considerations. Equation 1.1 shows the quadratic dependence of the dynamic power on the supply voltage of the circuits. Thus, the operating voltage must be decreased in order to restrict the dynamic power consumption in the circuit. According to ITRS03, the operating voltage decreases about 20% per technology node in order to keep the dynamic power consumption of the circuits in check. Table 1.1 shows the projected power trends over the next five years. It can be seen that although the supply voltage will decrease,

Table 1.1. Projected power trends (ITRS 2003)

Performance factor	2004	2005	2006	2007	2008
Supply voltage (V)	0.9	0.9	0.9	0.8	0.8
Power consumption (W)	158	167	180	189	200
Battery power (W)	2.2	2.3	2.4	2.5	2.6

the total power consumption in the circuit will continue to increase. This is because the increasing number of components within a fixed chip area will give rise to increasing component densities. The power consumption in the batteries which power these devices will correspondingly increase.

$$P_{cap} = C_L E(sw) V_{DD}^2 f \tag{1.1}$$

where C_L is the lumped node capacitance, $E(sw)$ is the switching activity at the node, V_{DD} is the supply voltage, and f is the clock frequency. However, the decreasing supply voltages will have adverse effects on both the delay as well as the noise sensitivity of the circuits. Equation 1.2 shows the inverse relation between the supply voltage V_{DD} and circuit delay t_D.

$$t_D = \frac{C_L}{2V_{DD}} \left(\frac{k_1}{\beta_p} + \frac{k_2}{\beta_n} \right) \tag{1.2}$$

where β_p and β_n are the gain factors of the p and n transistors respectively. k_1 and k_2 take values 1.5 and 2 for values of V_{DD} between 3 and 5 volts. While the increasing delay can be offset to some extent by lowering the threshold voltage of the transistors, the decreasing threshold voltage increases the noise sensitivity of the circuit. The noise sources are typically spread widely over the chip. The major noise source among these is the *interconnect noise*. This will be analyzed in detail.

The enormous number of components along with small device features increases the proximity between the different components of the design. Each individual component gives rise to a capacitance relative to the substrate. This is known as the *wire-to-substrate*

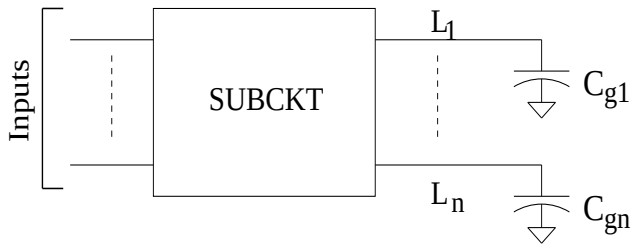


Figure 1.2. Subcircuit with wire-to-substrate capacitances

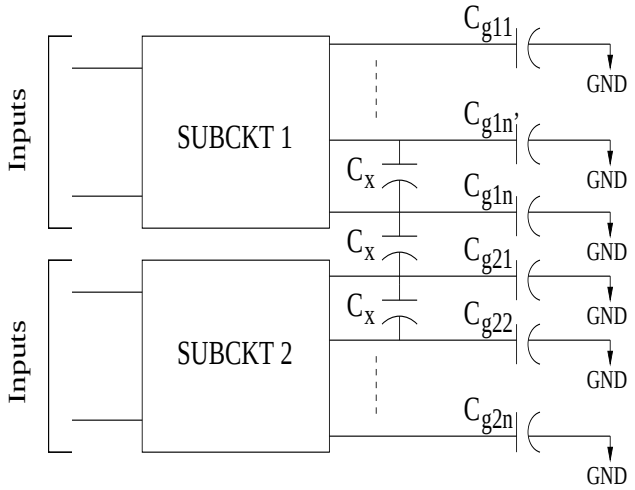


Figure 1.3. Two sub-circuits in close proximity

capacitance. For example, in Figure 1.2, each output line of the CMOS subcircuit has a capacitance C_g, relative to the ground. Thus, n-output lines have wire-to-substrate capacitances C_{g1} through C_{gn}.

On the other hand, a capacitance C_x may exist between

- Closely-located wires of a common subcircuit.
- Closely-located wires of different subcircuits which are in the vicinity of one another.

Such a capacitance is known as the *cross-coupling* capacitance. Both types of cross-coupling capacitances are shown in Figure 1.3. If the wire-to-substrate capacitance C_g is much higher than the cross-coupling capacitance C_x, then each line has a high drive strength and different lines do not influence each other in any way. However, at lower technology regimes, the thickness of the metal is increased, with respect to the spacing between the

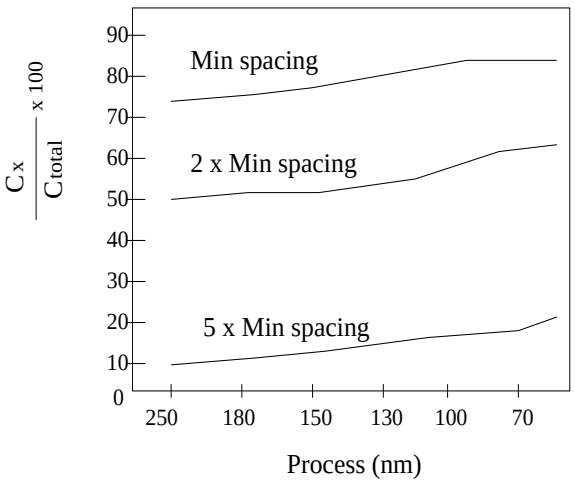


Figure 1.4. Cross-coupling capacitance as a dominant factor in nanometer technology (Reproduced from Kim et.al. [2])

lines, in order to maintain a low resistance for the lines [4]. The ratio of the cross-coupling capacitance to the wire-to-substrate capacitance could be as high as 3:1. As a result, the cross-coupling capacitance becomes significant as compared to the wire-to-substrate capacitance. Thus, previously unrelated events such as transitions on two closely-located lines, begin to influence one another. This phenomenon is known as *crosstalk* and is highly undesirable in digital circuits, since they have adverse effects on both the delay and the power consumption in the circuit, leading to signal integrity and reliability failures. This will cause the circuit to malfunction completely or worse still, function intermittently [4]. Figure 1.4 shows the projected dominance of cross-coupling capacitance with respect to the total wire capacitance with the scaling down of technology nodes [2].

The adverse effects of crosstalk are felt not only in the design field but in the test field as well. Due to increasing interaction between designers and testers and the advent of the *design for testability* concept, designers need to account for cross-coupling faults that may surface during the testing process of a circuit [5]. For example, a memory bank consisting of memory cells may be rejected because of coupling faults between adjacent memory cells

although this may have passed the design specification. Figure 1.5 shows a typical flow depicting the increasing interaction between design and test [6].

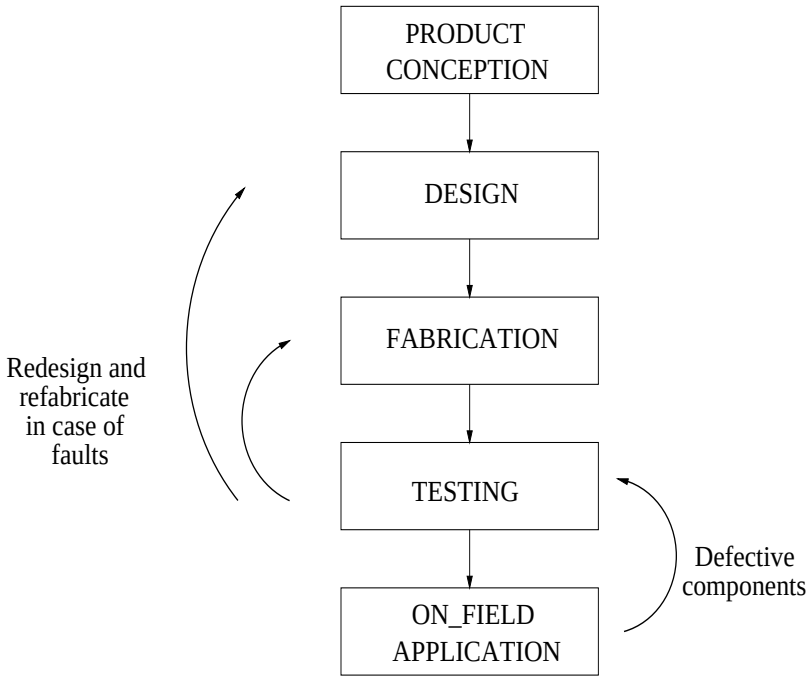


Figure 1.5. Interaction between design and test phases to eliminate coupling faults

1.1 Bus-based interconnects

With scaling down of technology, the interconnect assumes increasing importance in the design. The interconnect not only dominates the delay in the circuit but also consumes about 30% of the total dynamic power in the design. The most common interconnects are *buses* as bus-based interconnects reduces the number of connections compared to other interconnect styles. However, buses involve long lengths of wires running in parallel and in close vicinity of one another. With the decrease in the spacing between the lines compared to the thickness of the wires, the cross-coupling capacitance between the bus wires become comparable and often exceeds their wire-to-substrate capacitance.

A detailed depiction of the various capacitances related to the interconnects which run close to one another is provided in Figure 1.6. The area capacitance C_{area} and the capaci-

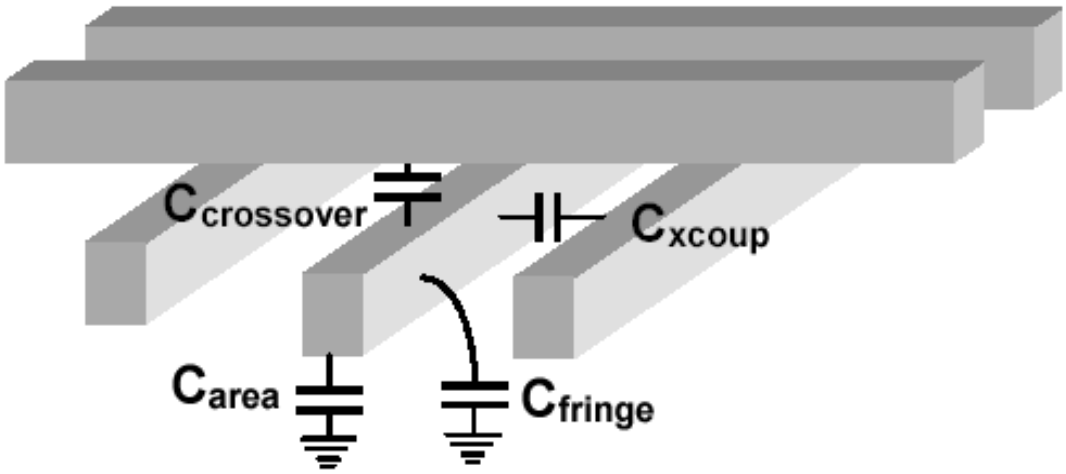


Figure 1.6. The various capacitances in a design

tance due to the fringe fields C_{fringe} together constitute the wire-to-substrate capacitance. The coupling capacitance C_{xcoup} exists between close wires running in the same plane. The coupling capacitance $C_{crossover}$ exists between close wires in different planes. When C_{xcoup} + $C_{crossover}$ becomes comparable to C_{area} + C_{fringe}, the *crosstalk* phenomenon comes into effect. Due to crosstalk, the circuit could produce erroneous results due to unwanted glitches on some of the lines. Alternatively, the circuit could malfunction due to induced delays on some of the lines which could violate their timing requirements. The lines which cause glitches and delays on other lines are known as *aggressors* while the lines which are affected by these aggressors are known as *victims*. The crosstalk effects between aggressors and victims are twofold and are stated as follows:

1. If the victim wire is at a steady state value (either logical '0' or logical '1') while the aggressor wire is switching (either from '0' to '1' or from '1' to '0'), it could induce an unwanted positive or negative spike on the victim wire.

2. If the victim wire is making a transition from '0' to '1' and the aggressor wire is also making a transition from '0' to '1', the transition of the victim wire will be *hastened* since the transition is being aided by the aggressor. On the other hand, if the victim

wire is making a transition from '0' to '1' and the aggressor is making a transition from '1' to '0', the transition of the victim wire will be *delayed* since the aggressor is now opposing the victim's transition. The transition of the victim wire from '1' to '0', with respect to the aggressor's transition, can be similarly analyzed.

1.2 ASIC design flow

Before going into the details of crosstalk, it is important to refresh the typical top-down design flow of Application-Specific Integrated Circuits (ASICs), in order to subsequently appreciate how the crosstalk problem can be tackled at different abstraction levels of a design. Figure 1.7 shows the top-down design flow of ASICs. Starting with the behavioral-level description of a design, typically in the form of a control data-flow graph (CDFG), high-level synthesis is performed. High-level synthesis (HLS) involves three steps namely, scheduling, allocation, and binding. During *scheduling*, individual operations in the CDFG are assigned to time steps in which they are to be executed. This is followed by *allocation* where resources are assigned to implement the different operations. Finally, *binding* maps individual operations to different instances of the resources. At the end of high-level synthesis, we get a register-transfer level (RTL) implementation of the behavioral description where the entire design is described in terms of a set of registers, functional units, and multiplexers along with data transfers between them. The interconnections between different components are accomplished through *buses*.

Logic synthesis is then performed on this RTL design during which each module is described in terms of its gate-level netlist. For example, a 2x1 multiplexer will be described in terms of an inverter, two AND gates, and an OR gate, as shown in Figure 1.8.

The entire gate-level netlist is then subjected to physical-level synthesis. Physical-level synthesis converts the gate-level design to the final layout through a sequence of steps namely, partitioning, floorplanning, placement, routing, and compaction [7]. *Partitioning* decomposes the gate-level circuit into sets of smaller gate-level circuits so that the smaller circuits may be designed independently and then combined. Such an approach speeds up

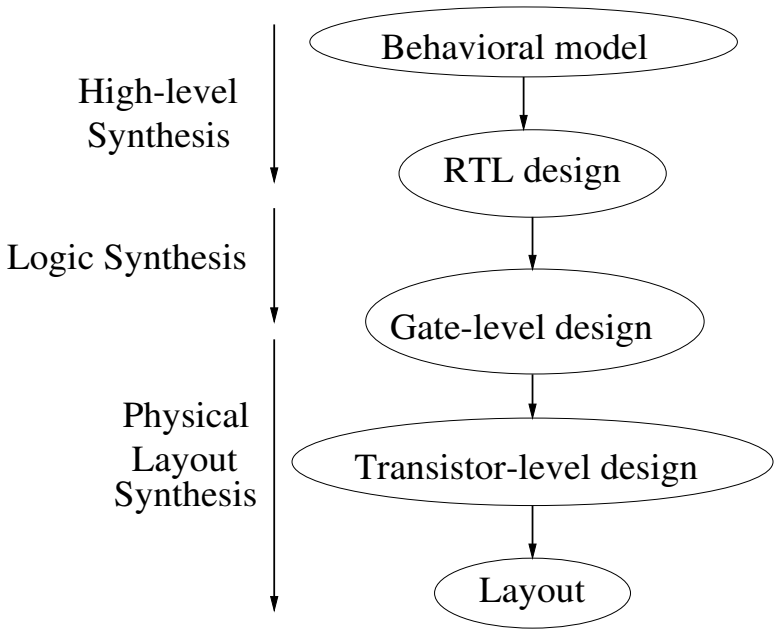


Figure 1.7. Top-down design flow

the design process. *Floorplanning* is the process of assigning approximate dimensions and shapes to the individual gate-level subcircuits to determine the relative positions between these in the final layout, usually with the objective of minimizing the layout area. The output of the floorplanning phase is then finalized in the *placement* phase where the circuit blocks are assigned to actual coordinates. In the *routing* phase, the pin connections between the placed blocks are completed using *bus wires*, with the objective of minimizing both delay and skew of critical signals. These effects will be discussed in greater detail in the subsequent chapters. Finally, *compaction* is performed to explore whether the layout area may be further reduced by eliminating any unused space between the modules.

1.3 High-level estimation: an advantage

Each design stage has its own models for crosstalk. Tradeoffs exist between the accuracy and complexity of these different models. The higher abstraction levels of the design yield much better *design-space exploration* than that of the lower levels of abstraction. In other words, evaluation of different solutions and movement from one solution to another

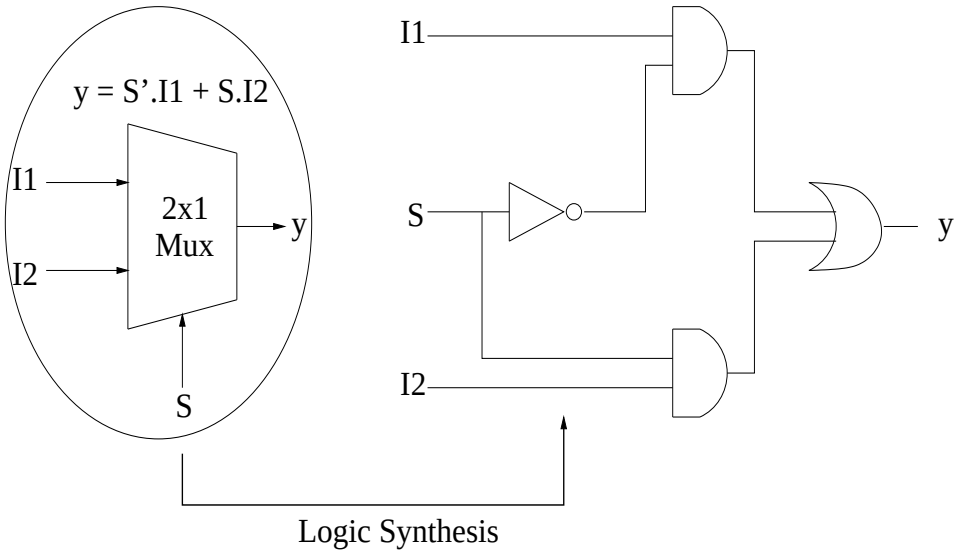


Figure 1.8. RT-level to gatel-level using logic synthesis

takes very little time and cost at the higher levels. For instance, analytical expressions for dynamic power estimation are preferred over simulation since the latter is expensive and takes unreasonable amounts of time in modern designs containing millions of transistors and wires. The cost involved in detecting errors and correcting them increases by a factor of 10 between each abstraction level as we move top-down in the ASIC design flow [8]. Thus, detecting crosstalk-sensitivity at the gate-level is 10 times more expensive than detecting it at the RT-level. Crosstalk minimization involving the physical synthesis steps of partitioning, placement, and routing, is likely to be ten times more expensive than making architectural transformations to minimize it before generating the gate-level netlist. This motivates us to think of new techniques to predict effects at the lower levels of abstraction from the higher levels of abstraction so as to minimize expensive errors at the lower levels. Crosstalk is a problem which could create design closure problems if detected in the later stages of the design, once most of the layout has already been fixed, since there is very little room to make changes in the placement and routing. Thus, there arises a critical need to design a fast and reasonably accurate *crosstalk noise estimator* to address the crosstalk problem in the earlier design stages.

However, the crosstalk noise problem is closely related to a lot of lower-level parameters which cannot be accurately determined at the higher abstraction levels. Some of these are as follows.

1. The positions of the modules, relative to one another.

2. The route of each wire.

3. The aggressors of a given wire in both the horizontal and the vertical planes.

For example, (1) cannot be determined before floorplanning has been performed. Similarly, (2) and (3) cannot be determined before routing has been performed. Thus, high-level crosstalk estimation becomes a hard problem and motivates us to devise techniques to integrate accurate information from the lower abstraction levels with the analytical estimation process at the higher abstraction levels.

In particular, statistical estimation methods have been shown to be very powerful while estimating dynamic power consumption in datapaths [9][10]. The main advantage of statistical estimation techniques lie in their fast execution times without loss of accuracy. Ramprasad et.al. [11] and Satyanarayana and Parhi [12] further demonstrate the usefulness of using word-level statistical parameters during high-level power estimation methods. The details of these approaches will be discussed in the next chapter.

1.4 Proposed approach to the crosstalk estimation problem

The notion of crosstalk is gradually changing, as will be amply demonstrated in the following chapter, from a *static* phenomenon which depends on a set of fixed parameters to a *dynamic* phenomenon which depends on the signal values of the neighboring wires at runtime. In this work, we treat the crosstalk phenomenon as a *pattern-dependent* phenomenon. In other words, whether or not an aggressor wire will cause crosstalk on a victim wire will depend on the relative signal values of the aggressor and victim wires. An aggressor will cause a crosstalk event on a victim if

1. The aggressor is switching while the victim is in a steady-state.

2. The aggressor as well as the victim are switching.

1.4.1 Intra-bus crosstalk estimation

The first type of events will cause crosstalk *spikes* or *glitches* on the victim. The second type will cause crosstalk *delays* on the victim, depending on the relative directions in which the aggressor and victim are switching. Thus, the crosstalk effect a victim experiences essentially depends on the values on the victim and its aggressor lines, over successive intervals of time. In other words, the crosstalk effect depends on the input data stream on the bus. Given an input data stream, we can compute the total number of crosstalk events that a victim line experiences. If the length of the data stream is known, we can associate a *crosstalk probability* with each victim line which is a direct measure of it's susceptibility to crosstalk.

To estimate the crosstalk probability, we enumerate the crosstalk-producing patterns on the victim, using the general aggressor-victim configuration of two adjacent aggressors on either side of a victim, shown in Figure 1.9. We propose a simple *stream-based estimator* that computes the crosstalk probability of a victim line in the bus. Given a data stream, the stream-based technique simply counts the total number of crosstalk-producing patterns and divides it by the length of the data stream to compute the crosstalk probability of a victim.

However, dealing with data streams is cumbersome and time-consuming. To speed up the estimation time, we propose a *word-level* statistical crosstalk estimator. The input to this estimator is a set of word-level statistical parameters (mean, standard deviation, and lag-1 temporal correlation coefficient) of the input data stream, instead of the stream itself. The output of the estimator is the bit-level crosstalk probability of each line of the bus.

Although the speedup obtained in the estimation time is high, the time complexity is still exponential, with respect to the bus width. In order to alleviate this problem, we use a

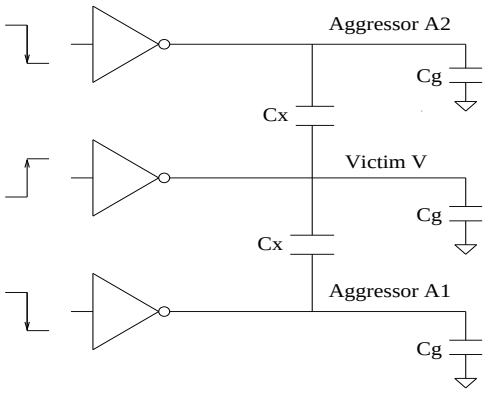


Figure 1.9. Aggressor-victim simulation circuit

circular right shift procedure that modifies the estimator to make the estimation time linear with respect to the bus-width.

We implemented the proposed approaches and validated them for several different data environments, modeled as ARMA signals, as well as bus-widths ranging from 8 bits to 32 bits. In other words, the proposed techniques are not restricted to a particular data stream but rather are designed to handle any data stream from specified data environments. We establish the exact match in the crosstalk probabilities estimated by the proposed stream-based estimator and that obtained through HSPICE simulations. At the same time, we demonstrate the faster speed of the stream-based estimation compared to HSPICE. Subsequently, we use the stream-based estimation as the basis for comparing the accuracy and speed of the proposed statistical estimation techniques.

1.4.2 Inter-bus crosstalk estimation

We extend the proposed word-level statistical estimator of intra-bus crosstalk, to measure crosstalk between different wires at the layout-level. While dealing with layouts, it becomes imperative to obtain accurate information about the placement of modules in the layout and the routes of the wires connecting different modules. Hence, we integrate a floorplanner and global router in our estimation flow which enables us to get the place-and-route information. We then form *composite buses* inside routing areas, using wires which

are located in the vicinity of one another. We perform RT-Level profiling on the design to get the word-level statistics on these composite buses. We then employ our statistical estimators to compute the crosstalk probabilities on each of these bus lines. Additionally, we analytically estimate the maximum noise pulse amplitude on a given victim wire in the layout using the information from the global router. Thus, we obtain the susceptibility of each victim both in terms of the total number of crosstalk events it is subjected to by its aggressors as well as the maximum noise amplitude it experiences.

Starting with the behavioral description of a design in terms of a data flow graph, we use *AUDI*, a high-level synthesis system developed by our research group at USF, to generate the RT-level netlist for each design. Using the Cadence NCLaunch simulator, we create our own RT-level profiler to obtain the data values on individual wires in the design at the register-transfer level. Using a pre-characterized cell library generated using Cadence icfb, we use the Cadence Silicon Ensemble synthesis tool to perform floorplanning and global routing. The target technology used in 0.35μ CMOS technology.

1.5 Optimization for crosstalk

We use the crosstalk susceptibility information of each victim net to explore the binding space and search for crosstalk-aware binding solutions during high-level synthesis. We use a modified clique-partitioning algorithm that takes crosstalk activity into account while selecting edges to bind to a given register. Comparisons with the regular binding solutions show crosstalk activity reductions at the register outputs.

1.6 Organization

The rest of the dissertation is organized as follows. Chapter 2 will discuss some of the existing works in the literature that have targeted to solve the crosstalk estimation and optimization problem and have elucidated the advantages of using statistical models in high-level estimation procedures for ASICs. Chapter 3 will introduce our approach to the intra-bus crosstalk estimation problem. In this chapter, we explain the *enumerative* statistical

estimator along with the *stream-based* estimator which we use as the basis for comparison. Chapter 4 will demonstrate the modification of the statistical enumerative technique to get the statistical *non-enumerative* technique and obtain two orders of speedup in the runtime of the estimation process. Chapter 5 will explain how we formulate the *inter-bus* crosstalk estimation problem so as to integrate the intra-bus crosstalk estimator along with layout-level information of the circuit and determine which wires, at the physical-level, are most affected by crosstalk. Chapter 6 will propose a crosstalk optimization procedure, based on the crosstalk-susceptibility information provided by the estimators. The experimental setups along with the results and their analysis will be presented at the end of each chapter. Finally, Chapter 7 will present conclusions and the directions for future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

The crosstalk problem has assumed increasing importance over the last few years. There has been research targeting estimation and optimization of crosstalk, at various levels of design abstraction. In this chapter, we discuss this research. In particular, we discuss the various models which have been used to capture the crosstalk effect in circuits as well as the key ideas used in crosstalk optimization. Special emphasis is given to statistical modeling of power consumption and its advantages as compared to other models since *statistical modeling* is central to our techniques. We also demonstrate the novelty of our approaches and how they fit into the overall ASIC synthesis flow.

2.1 Layout-level crosstalk estimation

In [13], an efficient technique for estimating the maximum coupled noise for on-chip interconnects is presented. This technique is a direct application of control theory. The transfer function function for the circuit with the aggressor and victim nets is computed. An input voltage in the form of a finite ramp is applied to the aggressor net. The final value theorem is applied to the victim net in order to get an upper bound on the coupled noise on the victim. This technique suffers from the two following limitations:

1. Since the noise on the victim net is dependent on the slope of the aggressor voltage, the noise on the victim can increase in an unbounded fashion if the slew rate of the aggressor voltage is very fast.

2. This technique does not account for the dependence of the coupling noise on the wire-to-substrate capacitances of the aggressor net and the victim net.

Hence, Kuhlmann et.al. [14] improve on the Devgan metric to accurately estimate the coupling noise, based on the sink capacitances of the victim and the aggressor, the coupling capacitance between them, and the rise time of the aggressor net. By efficiently solving the small signal model equations for the victim and aggressor nets, the estimation complexity is improved.

Various timing and glitch detection issues are analyzed using the *charge-sharing model* [15]. In this model, the final voltage at any node in the circuit is expressed as a ratio of the total charge to the total capacitance. During a transition at a node, the nodal voltage is expressed as a function of the capacitors' currents flowing to the ground node. This model serves as a reference for crosstalk estimation heuristics because of its ability to model nodal voltages. Vittal et.al. leverage on this in their attempt to address the crosstalk problem at the layout level [16][17][18]. They make several novel modifications to the charge-sharing model and derive analytical expressions for the coupled integral as well as an upper bound on the amplitude of the noise voltage. Some of the key differences with the *charge sharing model* are as follows:

- *Using dynamic noise margins as opposed to static noise margins used in the previous models.* Dynamic noise margins are more accurate as they account for the pulse amplitude as well as the pulse width of the noise [19][20][21][22]. Thus, a noise pulse of significant amplitude but very small duration is correctly determined to be harmless using dynamic noise analysis since the victim nets will not be driven to any erroneous value in such a short time.

- *Accounting for the dependence of the crosstalk noise on the drive strengths.* The previously used charge sharing model fails to account for this dependence. Suppose there is a victim wire v and two potential aggressor wires, w_1, with a small shared charge as well as small drive resistance, and w_2, with a larger shared charge and drive resistance. Further, let the frequency of transitions on w_1 be greater than that on w_2. Then, according to the charge sharing model, it is better to route v alongside w_1 but according to Vittal's model, it is better to route v alongside w_2. This is because in

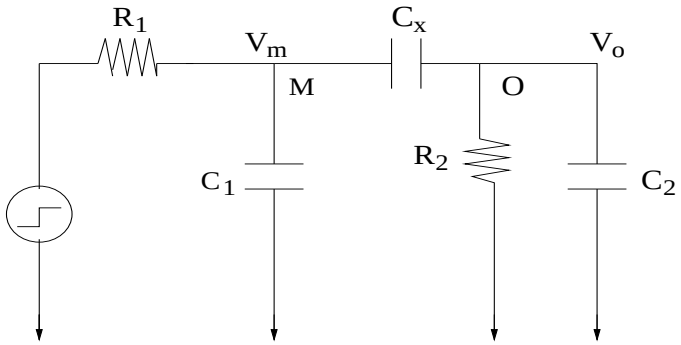


Figure 2.1. Equivalent circuit for computing crosstalk noise amplitude

Vittal's model, although the shared charge between the victim and aggressor is more, the total coupled noise on the victim is lesser due to smaller switching frequency of the aggressor.

- *Accounting for the dependence of crosstalk noise on the slew rate of the aggressor.* Vittal's model increases overlaps of the victim with nets which switch slowly in order to reduce the overall crosstalk on the victim. The tolerable overlap lengths are higher than the charge-sharing model resulting in more flexibility during routing.

Based on the Figure 2.1, the analytically computed bound on the crosstalk noise pulse is provided in Equation 2.1.

$$V_p = \frac{1}{1 + \frac{C_2}{C_X} + \frac{R_1}{R_2}(1 + \frac{C_1}{C_X})} \tag{2.1}$$

where R_1 and R_2 are the lumped resistances of the aggressor and victim nets respectively while C_1 and C_2 are the wire-to-ground capacitances of the aggressor and victim nets respectively. C_X is the coupling capacitance between the aggressor and victim. The maximum bound on the noise pulse amplitude is used to determine the noise-critical nets. The criticality of each net with respect to crosstalk is then incorporated into the cost function of a greedy channel router. The routing solutions are found to be optimized for crosstalk compared to the standard greedy channel routing solutions which do not consider crosstalk.

Vittal et.al. further provide an analytical expression for the crosstalk pulse width in [18], as shown in Equation 2.2.

$$V_{width} = \frac{K}{b_1} \tag{2.2}$$

where $K = \sum_{R_i \in P(o)} C_{X_i} R_i$ and $b_1 = \sum_{C_i \in C)} C_i R_{ii}$, where C_{X_i} is the sum of the coupling capacitances as seen from node i, $P(o)$ is the union of the victim drive resistance and the set of resistances in the path from root i to node o, C is the set of all capacitors, and R_{ii} is the resistance seen across capacitor C_i with all other capacitances open. The accuracy of the proposed estimation equations for the amplitude and the pulse width, with respect to HSPICE simulations, is found to be high with average errors less than 10% and these parameters are included in the cost function of optimization techniques namely, transistor sizing, wire ordering, and wire width optimization.

Typically, the crosstalk-induced delay in circuits is estimated by superposing the switching waveform of the victim and the noise waveform of the victim when it is *quiet*. However, this is found to underestimate the actual delay measure. To make the estimation process more accurate, a technique is presented by Tsai and Sadowska [23] that introduces the concept of *dynamic coupling noise*. This accounts for the dependence of the crosstalk noise amplitude and pulse width on the skew of the aggressor waveform.

2.2 Gate-level crosstalk estimation

A novel technique to minimize crosstalk in Programmable Logic Arrays (PLAs) is presented by Tien et.al. in [1]. PLAs are particularly susceptible to crosstalk as co-planar product lines run in close vicinity of one another for long distances. By simulation of the generic aggressor-victim configuration shown in Figure 1.9, the authors obtain the peak voltage of the victims. The target technology used by the authors is TSMC $0.35\mu m$. The coupling capacitance and peak voltages are summarized in Tables 2.1 and 2.2 respectively.

Table 2.1. Coupling capacitance of two wires for different overlapping lengths (reproduced from Tien et.al. [1])

Metal Layer		$100\mu m$	$300\mu m$	$500\mu m$	$1000\mu m$
M2	C_x	9.4fF	25.6fF	41.8fF	82.3fF
M2	C_g	4.9fF	13.7fF	22.4fF	44.4fF
M4	C_x	14.6fF	40.2fF	65.8fF	129.8fF
M4	C_g	3.3fF	9.0fF	14.6fF	28.8fF

Table 2.2. Crosstalk noise of the victim product line for different product lines (reproduced from Tien et.al. [21])

Metal Layer	# of aggressors	$100\mu m$	$300\mu m$	$500\mu m$	$1000\mu m$
M2	1	131.7mV	231.9mV	294.5mV	354.9mV
M2	2	260.4mV	453.2mV	568.0mV	690.6mV
M4	1	190.5mV	331.4mV	411.7mV	486.3mV
M4	2	375.1mV	655.0mV	813.2mV	990.0mV

Since several outputs may share a product term, the technique considers the output set of every product line i.e., the set of outputs which share that product line. A victim line is considered *crosstalk immune* if its output set is a subset of the output sets of its aggressors. This is because if each product term generated by a victim can be generated different aggressors, we can remove the victim line altogether. The authors find good orderings of both the product lines as well as the input-output lines in order to maximize the number of *crosstalk-immune* lines. The reordering also serves to reduce the wirelengths in the PLA. An example of the reduction in wirelength due to input-output reordering is shown in Figures 2.2 and 2.3.

Buyukashin and Najm [24] predict the interconnect effects during high-level power estimation by estimating the average interconnect length. Given a high-level description of the circuit, the authors use the well-known Rent's rule along with an estimation of the gate count in order to compute the interconnect length.

In terms of the design style, *domino* logic is very popular amongst contemporary designers. The performance of this design style is however, intricately connected to the quality of the interconnect design. In [2], the authors account for cross-coupling amongst inter-

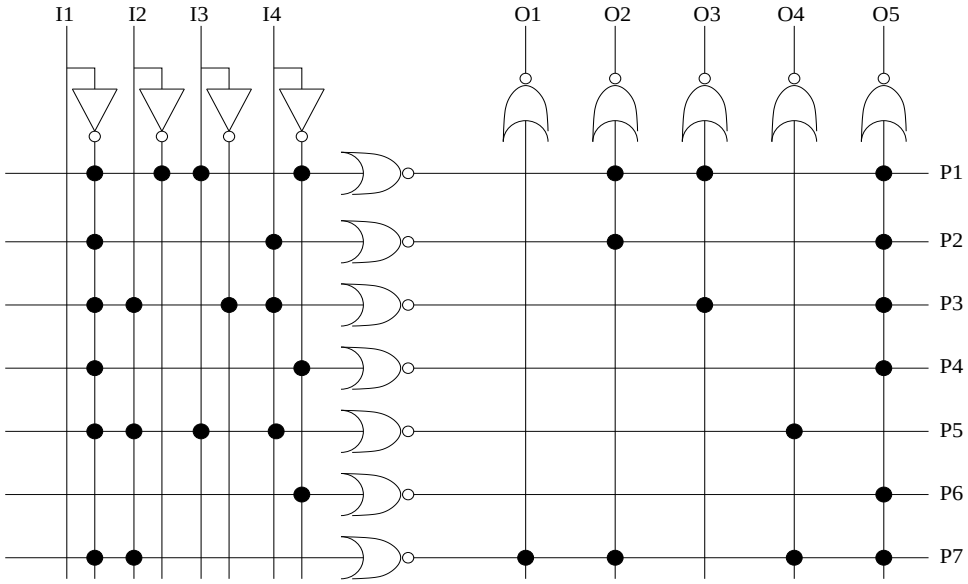


Figure 2.2. An example of a PLA (original configuration)

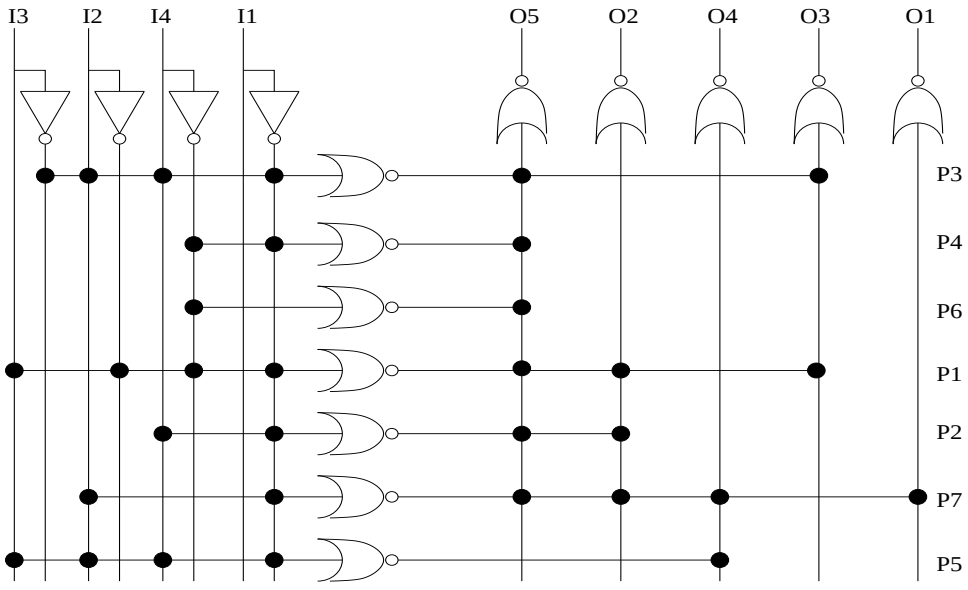


Figure 2.3. Reduction in wirelength using reordering

connect wires while formulating the cycle-averaged power model of the interconnect. The cross-coupling is added as a constraint while computing the maximum wire delay. The coupling power is then minimized using a simulated annealing approach by moving, swapping, and permuting the wires in the initial routing solution.

2.3 High-level estimation of circuit parameters

Research in literature exists that attempts to estimate circuit parameters such as power consumption, area, circuit delay, switching activity and even, coupling effects at the high-level. We summarize this work as a precursor to the estimation of crosstalk effects at the high-level.

In particular, power estimation as well as optimization during high-level synthesis, have been extensively researched [25][26][27][28][29][30][31]. Since the dynamic power is quadratically dependent on the supply voltage [32], a common technique is to use multiple supply voltages for different parts of the circuit. Components on the critical paths are activated using the higher voltages so as to strictly obey the timing constraints. The non-critical components are activated using a lower supply voltage so as to minimize the overall dynamic power [33][34]. A *profile-driven* approach to synthesize designs with minimum switching activity is presented in [35][36][37]. Various architectural transformations are used to optimize registers and interconnects during the synthesis process [38]. The design, specified as a data flow graph, is initially stimulated using user-supplied input patterns. Probes are inserted at various regions of the circuit specification to monitor event activity. A library of modules whose parameters are determined through characterization, is used in the layout generation phase. The layouts are generated using the Lager IV Silicon Compiler [39]. For validation of the power estimation procedure, switch-level models of the layout are extracted and validated using the IRSIM-CAP simulator.

Another technique is to dynamically alter the clock frequency to a component depending on whether it is part of the critical path or not. Once again components on the critical path are fed with a higher clock frequency while non-critical components are fed with a lower clock

frequency to minimize dynamic power In [40], the lower frequency operations are scheduled in earlier time steps while higher frequency operations are scheduled in later time steps. Subsequently, some of the higher frequency operations are regrouped with low frequency operations to obey the time constraints. Most of these approaches are time-constrained i.e., the throughput of the design is given. However, the multiple voltage approach has successfully been applied to the resource-constrained problem too [41][42]. Integer Linear Programming (ILP) models for energy and transient power minimization during the synthesis process have been proposed in [43]. The cycle power function (CPF) expressed in terms of the average power consumption and peak power differential accurately captures the power characteristics of data flow graphs in such multiple voltage configurations. Further, the authors propose a technique to modify the usually non-linear nature of the cycle power function so as to use ILP solutions on it [44]. Another approach to reduce the dynamic power consumption by reducing the switched capacitance inside different modules is presented in [45].

In [46], transition density [47] is used as a high-level metric for the switching activity in digital circuits. The transition density is defined as the average rate of switching at a circuit node. The average power consumption is approximately half the total power consumption. Thus, from Equation 1.1,

$$P_{cap,avg} = \frac{1}{2} C_L E(sw) V_{DD}^2 f = \frac{1}{2} C_L V_{DD}^2 * \frac{E(sw)}{T} \tag{2.3}$$

where $E(sw)$ is the total number of transitions over an interval T. The limit $\lim_{T\to\infty} \frac{E(sw)}{T}$ is referred to as the transition density. Using a stochastic model of the binary signals at the nodes, the technique propagates the transition density through the circuit modules. This is particularly useful for large circuits which would otherwise need extensive simulation with large vector streams to compute the switching activity at the various nodes.

Concepts from research areas as diverse as economics and mechanics have been incorporated into electronic design automation targeting power optimization and area optimization. In [48][49], the authors use the auction-based non-cooperative game theory based on Nash

equilibrium [50] to optimize power during behavioral synthesis. On the other hand, in [51], the authors use the concept of the mechanical spring force to minimize the area of a circuit for a given latency.

The high-level estimate of the gate-count is used by Nemani and Najm [52] as a high-level measure of the circuit area. The gate-count is estimated by converting multiple-output Boolean functions implemented by the circuit into an equivalent single-output function and then using the *on-sets* i.e., the minterms for which the function is true and *off-sets* i.e., the minterms for which the function is false of the new function, to determine the possible sharing of gates in the original multi-output function. By considering sharing, the minimum number of gates and thus, the minimum area is estimated. However, the technique is currently restricted to combinational circuits and does not work too well for circuits which contain large arrays of XOR gates.

In [53], the authors estimate the power consumption at the RT-level of a design by using *entropy* as a measure of the average switching activity in the circuit. The entropy is defined as the information-carrying capacity of a random variable. The entropy plot of a Boolean variable is a perfect bell-shaped curve, as shown in Figure 2.4. A Boolean variable with probability $p = 0.5$ has a value of '1' 50% of the time. Thus, it can make the maximum number of transitions and can carry the most information. The technique associates an entropy with each Boolean variable in a Boolean function to be implemented by the circuit. Thus, each function has a set of input and output entropies. From these, the average entropy at every node in the circuit can be determined depending on whether it serves as an input node or an output node or both. The average power consumption is then approximated as a product of the activity and the area. At the high-level, the area is estimated to be proportional to the number of don't-care terms in the Boolean function.

Further, the authors use *petri-nets* in order to model the real delay of both logic gates and interconnects in [54][55]. The real delay values are utilized in computing the overall switching activity in the circuit. Any given logic circuit is initially modeled as a gate-signal graph which is then transformed into a hierarchically colored hardware petri-net and

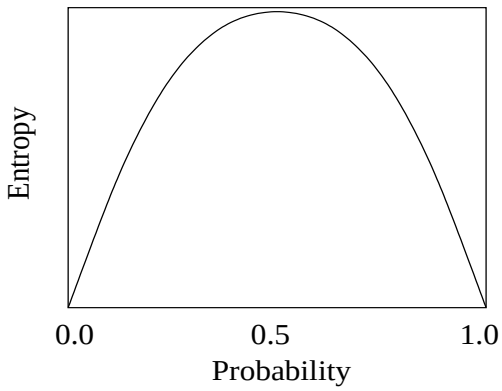


Figure 2.4. Bell-shaped curve of entropy for boolean variables

simulated to estimate the switching activity. Yet another approach to estimate switching activity is presented in [56] which uses the Bayesian network model from statistics. In this, Bayesian networks are used to capture complex conditional dependencies between different nodes in a circuit. The temporal and spatial correlations of the switching activity at a node is expressed as a switching probability model. These switching probabilities are propagated very efficiently through the Bayesian network.

The dependence of crosstalk-induced delays in buses, on the input data to the bus, has been analyzed in [57]. Each pattern has an associated delay. Using a characterization technique, various delay-groups are formed such that every data pattern to be transmitted over the bus belongs to one of those groups. The technique uses a faster clock and dynamically controls the number of cycles required to transmit any given data pattern, based on its delay value. Although this technique improves the performance by over 30%, it does not account for the crosstalk-induced spikes on the data bus. not hampered.

2.3.1 Worst-case crosstalk metrics

The *worst-case* crosstalk noise and delay metrics are often used in the analysis of circuit performance [58][59]. The proponents of this approach argue that if the circuit is designed for the worst-case performance, no fatal events will occur during its operation. The worst-case noise and delay are typically evaluated using one of the following two methods.

1. The noise resulting from several aggressors switching at the same time is determined using the *superposition* theorem i.e., considering only one aggressor to be active and the others to be silent at a time, and evaluating the peak noise on the victim from the active aggressor. The total noise is then a summation of the peak noises from each aggressor [13][60][61].

2. The total coupling noise on the victim is computed by assuming that the inputs to all the aggressors of the victim have the same arrival times. In other words, the aggressors switch simultaneously [62][63].

However, the worst-case metrics are pessimistic in nature. In reality, the worst-case rarely occurs and although the designs are made more robust, performance is often sacrificed. For example, if the clock frequency of a bus is set according to the worst-case delay on the bus, it may be quite low, thus making the system slower [57]. In reality, the worst case delay may occur very rarely. Thus, depending on the application, it may be better to set the frequency according to the average delay. This will result in a faster system which may still be susceptible to the worst-case crosstalk. Such an approach is taken in [64] where the *instability periods* of the aggressors of a given victim are computed from:

- The instability periods of the primary outputs given by the designers.

- The gate propagation delays computed using timing analysis.

From these, it is possible to compute the time in the clock period at which a signal makes its transition. This transition interval of a signal is called its instability period. From the instability period of signals, the number of *active* aggressors for a given victim can be computed. This, also known as the *maximal noise configuration*, results in a more realistic estimate of the worst-case noise on the victim.

2.4 Crosstalk optimization

So far, we have looked at cross-coupling estimation techniques. The estimates obtained are used in several optimization procedures to minimize the crosstalk in circuits. We look at some of these optimization techniques.

2.4.1 Shielding

One of the simplest techniques to minimize crosstalk is to use *shield wires* in between lines which are highly susceptible to crosstalk. These shield wires are kept at zero potential, in order to act as an effective barrier between the victim and the aggressor, as shown in Figure 2.5. In the Figure, there is a shielding wire at zero potential between lines $l2$ and $l3$. Thus, these lines will not affect one another with respect to crosstalk. However, line $l2$ may still suffer from crosstalk due to line $l1$ since there is no shielding between them. However, the disadvantage of this technique is that it increases the area overhead due to the large number of shield wires which will be required between victims and aggressors in large designs.

To counter the area overhead problem, Saxena and Gupta [65] integrate the separate steps of *power routing* and *signal routing* to minimize the number of shields, while satisfying all shielding constraints.

2.4.2 Wire reordering

Another popular crosstalk minimization technique is wire reordering. Reordering is the *shuffling* of wires, thereby changing their adjacencies with respect to other wires and making them immune to crosstalk effects [66]. Figure 2.6 illustrates the original and reordered configurations of a bus. In the original configuration, it may be seen that lines $l1$, $l2$, and $l3$ are all susceptible to crosstalk. The transition on $l2$ is delayed due to the opposing transition on $l1$ while the transition on $l2$ induces a downward spike on $l3$. However, by reordering lines $l2$ and $l3$, the delay effect on $l3$ is cancelled due to the opposing transitions

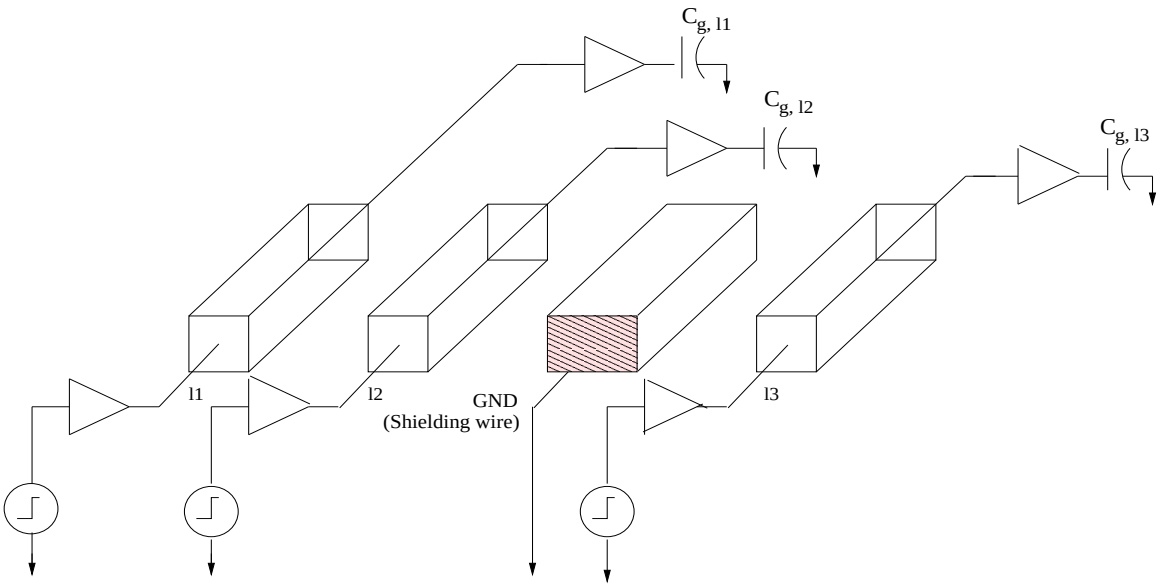


Figure 2.5. Shielding wire input victim and aggressor

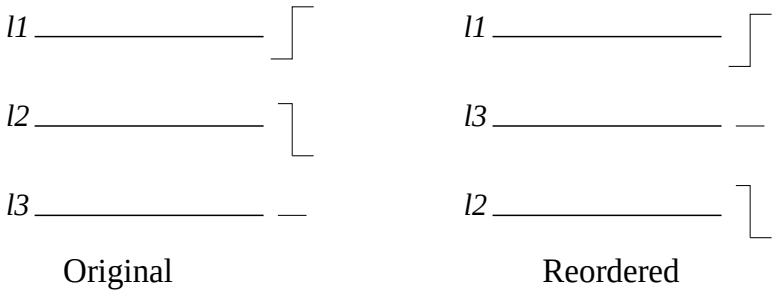


Figure 2.6. Eliminating crosstalk by bus reordering

on its aggressors $l1$ and $l2$. At the same time, $l2$ becomes farther removed from $l1$, making it immune to effects from $l1$.

2.4.3 Bus encoding

Crosstalk has been recognized as a data pattern-dependent phenomenon as opposed to its previous notion of being a static phenomenon. While bus reordering entails keeping the data values on the bus lines constant while changing the order of the bus lines, the bus encoding technique keeps the physical ordering of the bus lines constant while changing the data values on the bus lines according to an *encoding* scheme. This technique has been

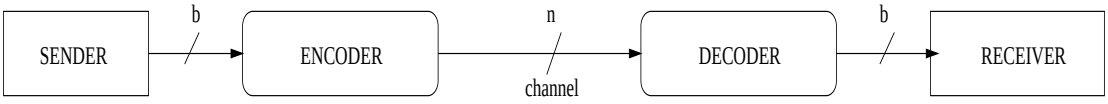


Figure 2.7. Communication chain model

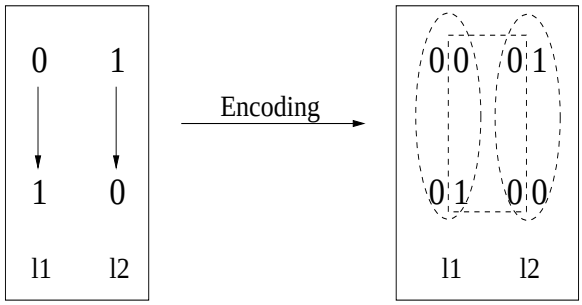


Figure 2.8. Eliminating delay with self-shielding codes

previously used for minimizing the glitch power activity on a data bus [67]. In the case of crosstalk, the data values are encoded so as to minimize crosstalk activity before being transmitted along the bus. At the receiving end, they are decoded back to their original values. Figure 2.7 illustrates the communication chain.

A bus encoding scheme to prevent crosstalk delay is presented in [68]. Here, adjacent lines are prevented from switching in opposite directions by using *self-shielding* codes, as shown in Figure 2.8. In the figure, lines $l1$ and $l2$ originally switch in opposite directions, causing crosstalk delay on either wire. However, by encoding each bit using its 2-bit binary value, adjacent lines are prevented from switching in opposite directions.

Another crosstalk minimization technique using capacitance optimization is proposed in [69]. This uses a placement technique that places the bus-lines non-uniformly, depending on the activity information of the lines. The technique minimizes crosstalk power without any increase in the complexity of the design. Several different algebraic, permutation-based, and probabilistic bus-encoding schemes targeting crosstalk minimization are discussed in [70][71][72][73].

2.5 Word-level statistical estimators

The inherent advantage of quick design-space exploration at the behavioral-level of designs motivates designers to develop estimators which can accurately predict the area, delay, and power parameters from high-levels of design abstraction. In particular, statistical, word-level estimators have been developed for dynamic power estimation [9] and transition activity estimation [11][12]. Since word-level statistics is central to the methods that we propose for crosstalk estimation, we look at some of the existing related estimators for power consumption closely in this section.

Landman and Rabaey [9] proposed a word-level statistical method using a *dual-bit type* (DBT) technique to estimate the dynamic power consumption in circuit datapaths at the architectural level. The dynamic power consumption is affected by the switching activity in the circuit inputs and edges, as shown in Equation 1.1. The dual-bit technique essentially captures the dependence of switching activity on the signal statistics using accurate "black-box" models for the digital circuits using a module characterization process. It accounts for both the uncorrelated lower-order bits as well as the heavily correlated higher-order bits in a word.

The dynamic power consumption P_D at any node in a circuit is directly proportional to the product of the load capacitance C_L and the switching activity $E(sw)$ at the node, as shown in Equation 1.1. Thus,

$$P_D \propto C_L E(sw) \tag{2.4}$$

The DBT-technique models the module capacitance and the switching activity inside every module separately. The DBT-model characterizes the *module capacitance* inside every module by using various types of inputs to the module. Thus, it generates a black-box model of the capacitance inside each module. With the assumption that the total capacitance inside a module is a function of its "size", the DBT-technique uses these parameterizable capacitance models to estimate the total capacitance inside a large module.

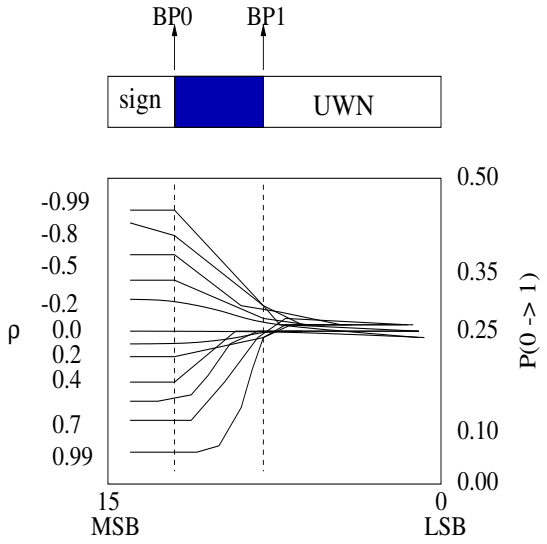


Figure 2.9. Different regions in a data word based on transition activity

The main difference between the DBT-model and previous models is that it accounts for correlated transitions as well as random transitions among data bits. For random word-level data values, the probability that a bit is either '0' or '1' is 0.5. Thus, the probability of a bit-transition from '0' to '1' is computed as follows:

$$P(0 \rightarrow 1) = P(0)P(1) = 0.5^2 = 0.25 \tag{2.5}$$

However, in a stream of word-level values, only some of the bits in a word are found to obey this relationship. Bits which obey the above relation constitute the *Uniform White Noise* (UWN) region in the data word. On the other hand, the higher order bits are more correlated. In fact, the highest bits are typically sign-bits in the data word and for most real-time signals, these bits have correlation close to 1.0. Such bits constitute the *sign* region of the data word. Thus, based on these observations from the characterization process, a data word can be split into three distinct regions, separated by two *breakpoints* $BP0$ and $BP1$, as shown in Figure 2.9. The positions of these breakpoints are dependent on the number of bits required to represent the numbers in the data stream with all the unused bits being part of the sign region.

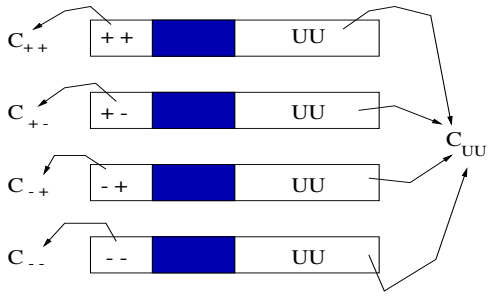


Figure 2.10. Capacitive coefficients for different sign-bit transitions

The breakpoints $BP0$ and $BP1$ can be expressed as functions of the word-level statistical parameters mean μ, standard deviation σ, and lag-1 temporal correlation coefficient ρ.

$$BP0 = log_2\sigma + \Delta BP0 \tag{2.6}$$

$$BP1 = log_2(|\mu| + 3\sigma) \tag{2.7}$$

To compute the capacitance switched by the different regions of the word, each module has different capacitive coefficients for the UWN and sign regions of the data word. The white noise region has a single capacitive coefficient C_{UU}. On the other hand, the capacitive coefficient for the signed region is a function of the set of capacitive coefficients corresponding to four distinct transitions of the sign bits, as shown in Figure 2.10. This leads to a series of capacitive coefficients which are stored as look-up-tables. Thus, the power analysis process is reduced to a series of table-lookups.

Ramprasad et.al. [11] present another word-level technique to estimate the bit-level transition activity in signals. They make a similar distinction between different regions of a data word, based on the lag-1 temporal correlation of the bits. In this method, the intermediate region between the white noise and the signed region is treated as a separate region where the correlation is assumed to linearly increase from the white noise value to the signed value, as shown in Equation 2.8.

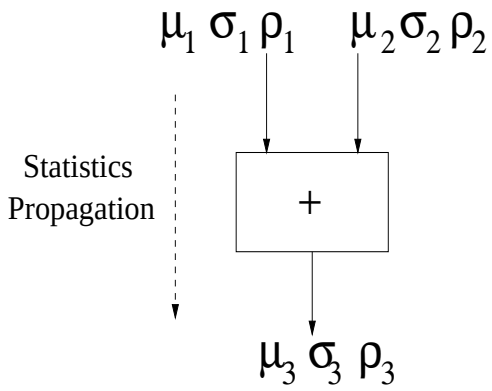


Figure 2.11. Statistics propagation in an adder

$$\begin{aligned}
\rho_i &= 0, (i < BP0) \\
\rho_i &= \frac{(i - BP0 + 1)\rho_{BP1}}{BP1 - BP0}, (BP0 \leq i \leq BP1 - 1) \\
\rho_i &= \rho_{BP1}, (i \geq BP1)
\end{aligned} \tag{2.8}$$

The authors present two methods to estimate the bit correlation ρ_i. The bit probability p_i is estimated from the probability distribution of the input. The transition activity t_i is then estimated using Equation 2.9.

$$t_i = 2p_i(1 - p_i)(1 - \rho_i) \tag{2.9}$$

When the bit-level correlation ρ_i is zero, the bits become temporally independent of one another and Equation 2.9 reduces to the product of $p(0)$ i.e., the probability of a bit being zero and $p(1)$ i.e., the probability of a bit being one. In [12], the authors extend the word-level estimation of switching activity to account for glitches in the signal. For signals with high correlation, this approach leads to faster computation times and greater accuracy than [11].

The advantage of using word-level statistics of signals as opposed to the actual signal values is that these statistics can be propagated from primary inputs to primary outputs very quickly. For example in the adder circuit in Figure 2.11, given the statistics μ_1, σ_1, ρ_1 and μ_2, σ_2, ρ_2 at the two inputs, we can directly compute the statistics μ_3, σ_3, ρ_3 at the output of the adder as follows:

$$\begin{aligned}
\mu_3 &= E[x_3(n)] = E[x_1(n) + x_2(n)] = \mu_1 + \mu_2 \\
\sigma_3^2 &= E[x_3^2(n)] - \mu_3^2 \\
&= \sigma_1^2 + \sigma_2^2 + 2E[x_1(n)x_2(n)] - 2\mu_1\mu_2 \\
\rho_3 &= \frac{E[x_3(n)x_3(n-1) - \mu_3^2}{\sigma_3^2} \\
&= \frac{\rho_1\sigma_1^2 + \rho_2\sigma_2^2 + E[x_2(n)x_1(n-1) + E[x_1(n)x_2(n-1) - 2\mu_1\mu_2}{\sigma_3^2}
\end{aligned} \tag{2.10}$$

The cross-covariances between independent signals simply reduces to the product of their means. For correlated signals, the cross-covariances between groups of primary inputs may be pre-computed and stored before beginning propagation of the statistics. Thus, the statistics at the output of every resource in the circuit may be computed analytically.

2.6 Data environments

The usefulness of any proposed technique in contemporary VLSI research is demonstrated by applying them to *data environments*. If a technique works well for a given data environment e.g., video, it essentially demonstrates the application of the technique to that domain. This is a vast improvement over techniques which are limited to specific data streams and cannot be applied to any given application domain.

The effectiveness of using *AutoRegressive Moving Average* (ARMA) models in generating data environments was demonstrated in [11]. ARMA models are commonly used to represent real-time signal domains such as speech and video.

An (N, M)-order autoregressive moving average model ($ARMA(N, M)$) can be represented as

$$x(n) = \Sigma d_i \gamma(n-i) + \Sigma a_i x(n-i) \tag{2.11}$$

where the signal $\gamma(n)$ is a white noise source whose mean value is zero, and $x(n)$ is the signal being generated [11][12]. The variable i gives the order of the ARMA model. For instance, i=1 gives the first order ARMA model. In this model, the signal $x(n)$ is related to its value in the previous instant. The coefficients d_i and a_i may chosen so as to minimize the mean-squared error. If the second term is zero, the signal $x(n)$ is purely dependent on the white noise input and is independent of the previous value $x(n-1)$. If the second term is non-zero, then $x(n)$ is dependent on the previous value. In this case, the temporal dependence is given by the temporal correlation coefficient ρ_x. The scaling factor of the white noise gives the mean μ_x while any additional term gives the standard deviation σ_x of the signal $x(n)$. Thus, the modified ARMA model is

$$x(n) = \mu_x \gamma(n) + \sigma_x + \rho_x x(n-1) \tag{2.12}$$

Thus, for a given set of word-level statistics μ_x, σ_x, ρ_x, we can generate a data stream corresponding to these statistics. This is useful while comparing our word-level statistical techniques against detailed HSPICE simulations and the stream-based estimation procedures.

2.7 Summary

To summarize, this chapter

1. proposed crosstalk estimators and metrics at various levels of design abstractions.

2. reviewed popular crosstalk optimization techniques.

3. summarized word-level statistical estimators which have been previously used to measure the transition activity at circuit nodes.

4. discussed the usefulness of data environment modeling and the techniques to model them.

From the literature search, our observation was that there is a dearth of techniques which attempt to tackle the crosstalk problem at the high-level. Moreover, there is no work that attempts to model the crosstalk problem statistically. This is the primary motivation for the crosstalk estimation and optimization techniques that are proposed in this work. The subsequent chapters present these techniques in detail.

CHAPTER 3

INTRA-BUS CROSSTALK ESTIMATION USING WORD-LEVEL STATISTICS

This chapter introduces our approach to the crosstalk estimation problem. It formally states the problem that we try to address and details the key contributions of the solutions that we propose. Two high-level techniques to estimate the probability of crosstalk events on signal lines of a system bus are presented: (1) Given an input data stream, the first technique simply estimates the number of crosstalk events on each line of the bus. The main drawback of this technique is that the execution time is proportional to the stream length. This is overcome by the second technique. (2) Given the word-level statistical parameters, namely mean, standard deviation, and lag-one temporal correlation coefficient, we estimate the bit-level crosstalk probability. Experimental results for data streams from different data environments, compared against detailed HSPICE simulations, are presented. The stream-based technique matches the HSPICE simulations exactly while average errors of less than 7% are obtained for the statistical enumerative technique. The crosstalk estimation time using the statistical method is significantly less than the stream-based technique and HSPICE simulations.

3.1 Modeling the crosstalk estimation problem

As discussed in the previous chapter, there is an acute need of high-level estimation techniques of crosstalk. The ability to predict details about the lower-levels of design abstraction like the physical-level, from the higher levels, provides tremendous leverage to the designer in exploring the design-space efficiently and generating an optimized design in the very first cut.

The estimation techniques proposed so far treat the amplitude and width of the noise pulse as the objective functions. Our approach differs from them in that we take a high-level view of the problem and treat all possible crosstalk phenomenon collectively as *crosstalk effects*. Based on the number of crosstalk events on every line of the bus, we evaluate a probability that indicates the amount of crosstalk activity on that line. The advantage of our approach is that it gives the designer an idea as to which bus lines are more susceptible to crosstalk. Based on this information, bus lines which are highly susceptible to crosstalk may then be reordered or encoded[68][72][1][69][57].

Formal statement of the problem : Given only the word-level statistics of the data on a system bus, the objective is to analytically estimate the crosstalk susceptibility of each bus line. The technique should be independent of the length of data streams on the bus so that the estimation process takes a constant time, irrespective of the data stream length.

3.2 Problem formulation

A wire having coupling capacitance with another wire may be treated as either a *victim* or an *aggressor* with respect to the other wire. If it is a victim, it means that transitions on the other wire will adversely affect the steady state as well as transitions on it. If it is an aggressor, it means that it will affect the other wire adversely. Figure 3.3 [1] shows the victim wire V entrapped between the aggressors *A1* and *A2*. However, the wire V could also act as an aggressor for both *A1* and *A2*.

The coupled capacitance between wires has an effect on both the delay and power dissipation of the victim line. If the victim is at a steady-state value and the aggressor is switching, it produces a spike on the victim in the direction in which the aggressor is switching. This has the effect of unwanted power dissipation on the victim line. In particular, if the victim develops a spike below its logic level '0' V_L or above its logic level '1' V_H, it is referred to as the *bootstrap noise*. If the victim is switching in the same direction as the aggressor, the latter reinforces the victim's transition, thereby hastening it.

Table 3.1. Possible crosstalk effects

Signal Transition on Aggressor	Signal Transition on Victim	Resultant Crosstalk effect
$0 \rightarrow 1$	$0 \rightarrow 0$	Upward spike
$1 \rightarrow 0$	$0 \rightarrow 0$	Bootstrap spike
$0 \rightarrow 1$	$1 \rightarrow 1$	Bootstrap spike
$1 \rightarrow 0$	$1 \rightarrow 1$	Downward spike
$0 \rightarrow 1$	$0 \rightarrow 1$	Vic transition hastened
$1 \rightarrow 0$	$0 \rightarrow 1$	Vic transition delayed
$1 \rightarrow 0$	$1 \rightarrow 0$	Vic transition hastened
$0 \rightarrow 1$	$1 \rightarrow 0$	Vic transition delayed

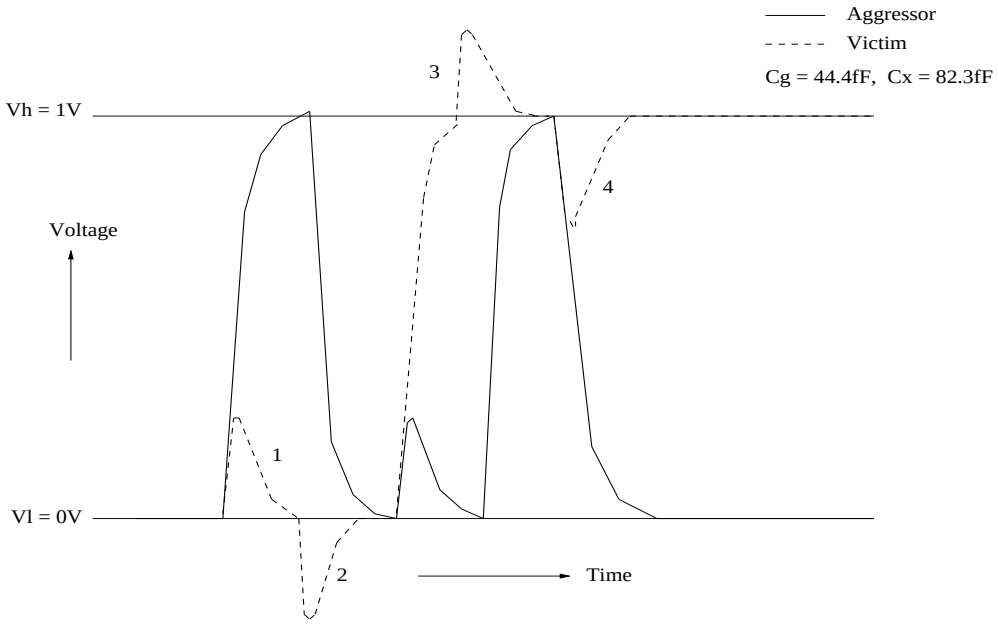


Figure 3.1. Crosstalk spike effects on victims

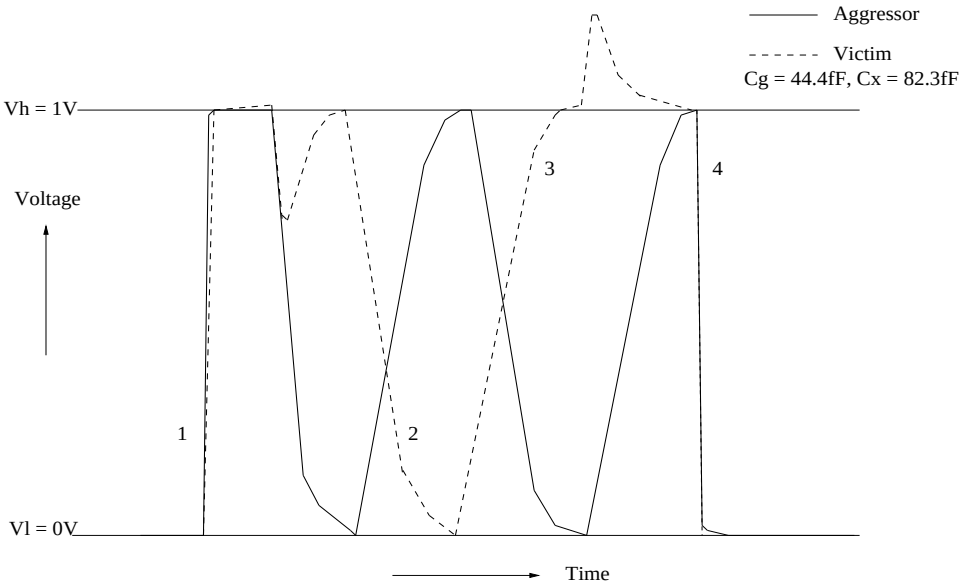


Figure 3.2. Crosstalk delay effects in victims

This effect is often ignored under the simplistic assumption that if adjacent lines switch in the same direction, the coupling capacitance between them is zero [74]. However, this could underestimate the actual delay of the victim [74]. If the victim is an input to a latch, this effect would cause violation in the *hold time* requirements of the latch, thereby upsetting the next state of the sequential circuit. Again, if the victim is switching in a direction opposite to that of the aggressor, the transition of the victim gets opposed by the aggressor, thereby delaying it. This would also affect the timing of the circuit adversely. Table 3.1 gives a summary of the possible crosstalk effects. In Figure 3.1, spikes 2 and 3 indicate bootstrap noise on the victim while 1 and 4 indicate upward and downward spikes respectively. In Figure 3.2, 1 and 4 indicate hastening of the victim's transitions while 2 and 3 indicate the delaying of the victim's transitions. These plots were obtained using HSPICE by simulating a circuit with one aggressor and one victim.

Key contributions of our work: The main contributions of our work are as follows:

1. We address the intra-bus crosstalk problem at the high-level. Given only word-level statistics of signals, we propose a statistical technique that estimates the amount of crosstalk such signals cause on the lines of a system bus.

2. We propose a new metric, namely the *probability of crosstalk events* for each line of the bus.

3. For verification of the proposed technique's results with physical-level results, we propose a stream-based technique that computes the total number of crosstalk events in a data stream that is an input to the bus.

4. Using detailed HSPICE simulation, we show that the stream-based technique is exactly as accurate and faster than HSPICE.

5. The proposed statistical technique is independent of data stream lengths. Thus, the runtimes are considerably reduced while accuracy is maintained.

The proposed stream-based technique that we use for verification simply counts the number of crosstalk-producing patterns in the input data stream. The total number of crosstalk events on each line of the bus gives us an estimate of the probability of such events on that line. On the other hand, the proposed statistical technique relies only on word-level statistical parameters of the input data stream, namely the mean, standard deviation, and lag-1 temporal correlation, to compute a bit-level crosstalk probability for each line of the bus. Hence, it is independent of the data stream length. The proposed statistical technique has been currently applied to estimate intra-bus crosstalk.

3.3 Proposed technique 1: stream-based crosstalk event estimator

The previous section illustrates that the crosstalk effects arise due to transitions of the *aggressors* with respect to the *victim* wires. Since we consider intra-bus crosstalk, it is reasonable to assume that all the lines of the bus are clocked at the same instant. Hence,

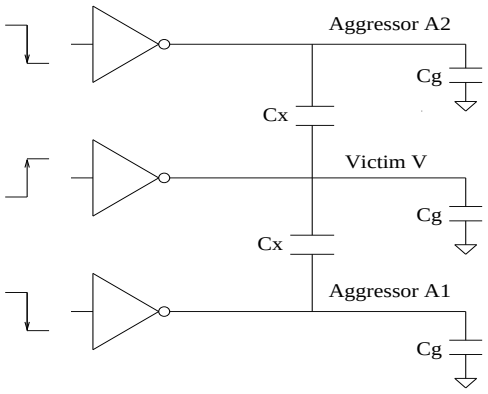


Figure 3.3. Aggressor-victim simulation circuit

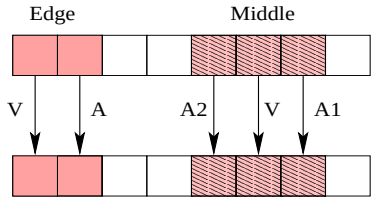


Figure 3.4. Checking bit transitions for crosstalk patterns

the aggressor-victim transitions translate to specific *bit patterns* in the input data stream to the bus.

The proposed stream-based estimator traces the input data stream for patterns which will cause crosstalk. The technique can be applied to any data environment. The *middle* lines in the bus suffer crosstalk effects from both their adjacent aggressors while the *edge* lines have only one aggressor which is responsible for crosstalk. Thus, the middle bits of the data word loaded onto the bus at every instant, are compared to the same bits of the previous data word in groups of three, namely $A_2\ V\ A_1$. On the other hand, the edge bits (both the MSB as well as the LSB) of the current word are compared to the previous word in groups of two namely $V\ A$ for the MSB and $A\ V$ for the LSB. Figure 3.4 illustrates the comparisons.

The procedure then counts the number of bit transitions on the aggressors that contribute towards crosstalk separately, for every line on the bus. Thus, at the end of the procedure, we have the total number of crosstalk events on every line on the bus. Since the

Table 3.2. HSPICE vs stream-based run times

Data word size (bits)	Length	HSPICE (time)	Stream-based (time)
8	1000	32s	25s
8	3000	129s	70s
16	1000	102s	26s
16	2000	147s	69s

length of the data stream is known, we obtain the probability p of a crosstalk event on any bus line using the following equation:

$$p = \frac{n}{l} \tag{3.1}$$

where n is the number of crosstalk events and l is the length of the data stream.

The accuracy of this method is compared against detailed HSPICE simulation. The spice netlist of the m-bit bus is simulated using the same input data stream. The total number of crosstalk events reported on each line of the bus by HSPICE, is matched with our approach. As expected, there is an exact one-to-one correspondence in the total number of crosstalk events obtained on each line of the bus by the two methods. However, our approach is faster than HSPICE because it reads the crosstalk events directly from the data stream without simulating it. Table 3.2 compares the run times of the two approaches for data streams of various lengths, generated using ARMA models [11]. All experiments were performed on a Sun Ultra 2 dual-processor workstation.

3.4 Proposed technique 2: statistical enumerative approach for crosstalk event estimation

The drawback of the stream-based technique is that it is dependent on the length of the data stream. To alleviate this, we propose a statistical technique that computes the crosstalk probability on every line of the bus, based on the statistical parameters of the data stream $x(n)$. This makes it independent of the data stream length.

The proposed approach makes use of some statistical parameters of the data stream which are mathematically defined as follows:

1. Mean of the stream *x(n)*

$$\mu = E[x(n)] \tag{3.2}$$

2. Standard deviation of the stream

$$\sigma = \sqrt{E[x^2(n)] - E^2[x(n)]}$$

$$= \sqrt{E[x^2(n)] - \mu^2} \tag{3.3}$$

3. Temporal (lag-1) correlation of the stream: This indicates how, at any instant, the current value $x(n)$ in the stream is related to the value $x(n-1)$ at the previous instant and is given by:

$$\rho = \frac{E(x(n) - \mu)(x(n-1) - \mu)}{E[(x(n) - \mu)^2]}$$

$$= \frac{E[x(n)x(n-1)] - \mu^2}{\sigma^2} \tag{3.4}$$

Let us assume that the signal $x(n)$ on the bus has a normal distribution. There is, however, no restriction on the distribution of $x(n)$. We just assume the normal distribution as a general case [12]. Typically, the probability distribution of $x(n)$ can be estimated from the ARMA signal generation models. In this work, we assume that the distribution of $x(n)$ is known beforehand. In the context of high-level power estimation using word-level statistics, Ramprasad et.al. [11] and Satyanarayana et.al [12] make similar assumptions about the data environments.

The probability p_i of the ith bit b_i in the data word is the probability that the ith line of the bus is a 1. This is given by:

$$p_i = Pr(x(n) \in \zeta_i)$$

$$= \sum_{j \in \zeta_i} \frac{1}{\sigma\sqrt{2\pi}} e^{-(j-\mu)^2/2\sigma^2} \tag{3.5}$$

where ζ_i is the set of all elements in ζ whose ith bit is 1.

From equation 4, it is evident that the temporal correlation of *x(n)* is dependent on the covariance $E[x(n)x(n-1)]$ term which is a statistical parameter, generally independent of the mean and variance of the stream. As an exception, if the data values in the stream are random, then the covariance term can be expressed as a square of the mean. But, for a highly correlated data stream, the covariance cannot be estimated from the mean and variance of the data.

It is thus difficult to accurately estimate the temporal characteristic of the data from other word-level statistics. Further, it is cumbersome to compute the same from bit-level information. Hence, we employ a *concatenation* procedure which enables us to continue work at the word-level, by transforming the temporal characteristic at the bit-level to the spatial characteristics of the concatenated streams.

Let us assume that the on-chip bus is m bits wide. Correspondingly, there are m bits in the data loaded onto the bus during successive clock cycles. If the data value at every instant $x(n)$ is concatenated with the value during the previous instant $x(n-1)$, we obtain a *compound* data word $X(n)$ of width $2m$. Figure 3.5 depicts the formation of the compound word when m is 8 bits wide.

The concatenated data stream $X(n)$ can then be expressed in terms of the component streams $x(n-1)$ and $x(n)$ using the following equation:

$$X(n) = 2^m x(n-1) + x(n) \tag{3.6}$$

We derive analytical equations to compute the mean and standard deviation of the compound stream $X(n)$ from the statistics of the original stream $x(n)$ as follows.

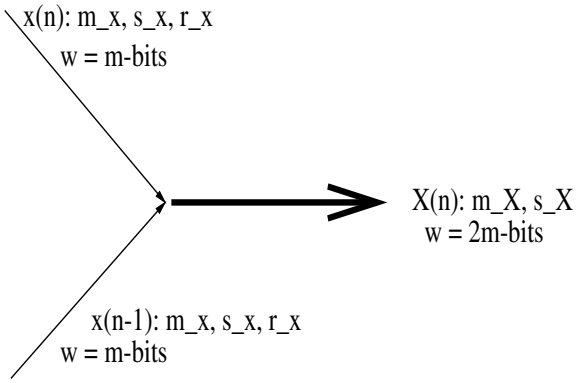


Figure 3.5. Concatenation

$$\begin{aligned} X(n) &= 2^m x(n-1) + x(n) \\ E[X(n)] &= 2^m E[x(n-1)] + E[x(n)] \\ \mu_X &= 2^m \mu_x + \mu_x \\ &= (2^m + 1)\mu_x \end{aligned} \tag{3.7}$$

$$\begin{aligned} \sigma_X^2 &= E[X^2(n)] - E^2[X(n)] \\ &= E[2^{2m}x^2(n-1) + x^2(n) + 2^{m+1}x(n-1)x(n)] - \mu_X^2 \\ &= E[2^{2m}x^2(n-1) + x^2(n) + 2^{m+1}x(n-1)x(n)] \\ &- 2^{2m}E^2[x(n-1)] - E^2[x(n)] - 2^{m+1}E[x(n-1)]E[x(n)] \\ &= 2^{2m}[E[x^2(n-1)] - E^2[x(n-1)]] + E[x^2(n)] - E^2[x(n)] \\ &+ 2^{m+1}[E[x(n-1)x(n)] - \mu^2] \\ &= (2^{2m}+1)\sigma_x^2 + 2^{m+1}[\rho_x\sigma_x^2 + \mu_x^2 - \mu_x^2] \\ &= (2^{2m} + 1 + 2^{m+1}\rho_x)\sigma_x^2 \\ \sigma_X &= \sigma_x\sqrt{2^{2m} + 1 + 2^{m+1}\rho_x} \end{aligned} \tag{3.8}$$

From equation (3.6), we observe that concatenation is a linear operation involving a multiplication and an addition operation. Any linear operations on normal distributions results in another normal distribution with a different mean and standard deviation [75][76]. Thus, the concatenated data stream $X(n)$ has a normal distribution with mean μ_X and standard deviation σ_X. We then utilize these parameters to compute the crosstalk probability for the middle lines of the bus.

3.4.1 Crosstalk for middle lines

We assume only first order crosstalk effects in our work in order to demonstrate the technique. This means that any victim is affected by only its immediately adjacent aggressors. Thus, within the bus, there are two aggressors *A2* and *A1* affecting any victim *V* in the middle. *A2* and *A1* are immediately adjacent to *V* in the sequence *A2 V A1*. However, the proposed technique can easily be extended to incorporate higher order effects. For first order effects, the victim lines experience crosstalk when either one of the aggressors switch while the other one is steady or when both the aggressors switch in the same direction. Opposite transitions on adjoining aggressors of any victim line nullify the crosstalk effect on the victim.

The concatenated word $X(n)$ *(b15 - b0)* has *two* crosstalk *windows*, as shown in Figure 3.6. The first is in the $x(n)$ region while the second is in the $x(n-1)$ region. It can be seen that these two windows are always separated by a constant distance, equal to the width m of the bus. The concatenated $X(n)$ is referred to as a *crosstalk template* which is defined as a *2m*-bit word where *six* of the bits are substituted with a given crosstalk pattern of interest.

The proposed technique involves substituting crosstalk patterns in the current crosstalk windows. For first order crosstalk effects, the crosstalk patterns are of length 6 bits, with each window having 3 bits. A typical crosstalk pattern is *001111* since it causes aggressor *A2* to switch from '0' to '1' while aggressor *A1* is fixed at '1' and victim *V* is switching from '0' to '1'. This pattern has the effect of hastening the transition on *V*. Similarly, *101001* is a crosstalk pattern while *110011* is not. Each valid crosstalk pattern of interest is referred

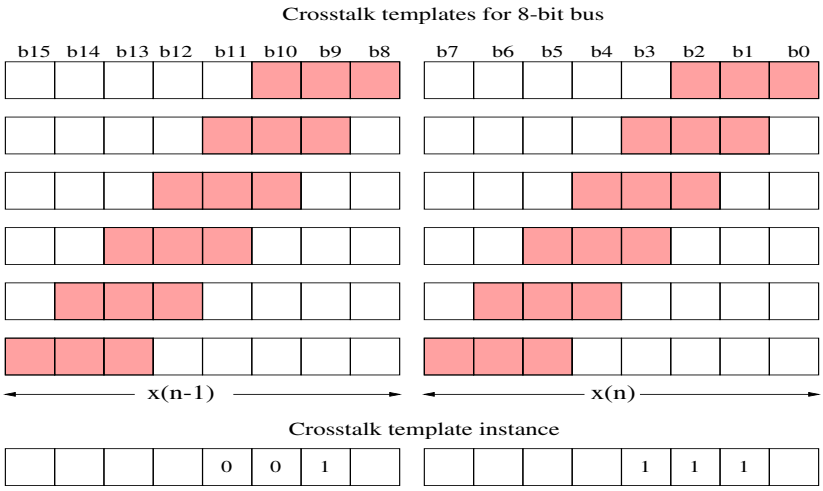


Figure 3.6. Bit-level crosstalk templates for 8-bit bus

to as a *crosstalk template instance*. For first-order crosstalk effects on every bus line, there are 40 template instances which cover all possible crosstalk effects listed in Table 3.1.

Once the crosstalk windows are filled with a given crosstalk pattern, the remaining bits in $X(n)$ are iteratively filled for all possible combinations. Each of those word-level values of $X(n)$ is a crosstalk-producing value for the current template instance.

Next, we compute the probability that such crosstalk-producing values are present in the data stream. The probability that a discrete valued variable $X(n)$ assumes a certain value in a given probability distribution is obtained by substituting the value in the probability distribution function itself [75]. Thus, in the concatenated, normally distributed data stream $X(n)$, the probability that a particular crosstalk-producing value j is present, is given by

$$p_j = P(X = j) = \frac{1}{\sigma_X \sqrt{2\pi}} e^{-(j-\mu_X)^2/2\sigma_X^2} \tag{3.9}$$

The values of the mean μ_X and standard deviation σ_X of $X(n)$ are analytically computed from equations (3.7) and (3.8). Thus, for fixed positions of the crosstalk windows, say *b0-b2* and *b8-b10*, we are able to evaluate the probability of crosstalk-producing values by summing up the probabilities of all crosstalk-producing words in $X(n)$ for every crosstalk template

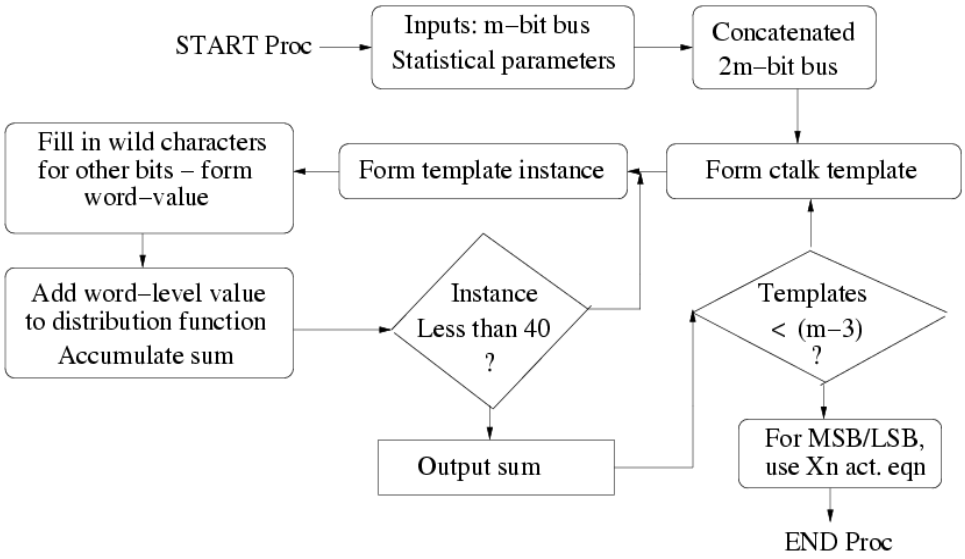


Figure 3.7. Procedural flow

instance. This gives the probability that the bit *b1* in the *m*-bit data word experiences crosstalk. Mathematically,

$$p_{b1} = \sum_{j \in templ_inst} p_j \tag{3.10}$$

The crosstalk windows then slide one position to the left namely *b1-b3* and *b9-b11* to similarly evaluate the crosstalk probability of the bit *b2*. This procedure keeps repeating until the windows reach the MSB positions, as shown in Figure 3.6. After the procedure is complete, we obtain a crosstalk probability for each of the middle lines in the bus. Figure3.7 gives a summary of the technique.

3.4.2 Crosstalk for edge lines

As stated previously, the MSB and LSB lines of the bus have only one aggressor each. They experience crosstalk whenever their aggressor undergoes a transition. The transition could be in either direction. If the edge line is in a steady state, it will dissipate power due to crosstalk spikes. If it is switching, it will experience early transitions or delays depending on whether it is switching in the same or in the opposite direction as the aggressor, respectively.

Table 3.3. Data environments

Bit-width	Environment	ARMA equation
8	SIG 1	x(n) = 50γ(n) - 10
8	SIG 2	x(n) = 30γ(n) + 0.5x(n-1)
8	SIG 3	x(n) = 75γ(n)
8	SIG 4	x(n) = 50γ(n) + 0.7x(n-1)
10	SIG 1	x(n) = 250γ(n) + 100
10	SIG 2	x(n) = 200γ(n) + 0.5x(n-1)

Thus, the probability of crosstalk on the edge wires is the same as the transition probability of their aggressors. From [11], the transition activity of the ith bit is given by

$$t_i = 2p_i(1 - p_i)(1 - \rho_i) \tag{3.11}$$

where p_i and ρ_i are the bit probability and bit correlation of the ith bit respectively. Both these parameters can be estimated using analytical techniques presented in [11]. Thus, using equation 3.11, we obtain the crosstalk probability of the edge bits from the word level statistics of the data stream.

3.5 Experimental results

As the proposed stream-based estimation is as accurate and faster than HSPICE in terms of measuring crosstalk events, we replace HSPICE with our stream-based estimator as the basis for comparing the quality of the subsequently proposed statistical crosstalk estimators.

Table 3.3 shows the data environments modeled using ARMA models. Such ARMA models are often used to represent speech and video signals. The proposed approach is general enough to handle various data environments. This means that it will work on all data streams from such environments. The white noise factor γ(n) has a standard normal distribution.

Tables 3.4 and 3.5 compare the crosstalk probabilities computed by the stream-based and statistical enumeration procedures for $m = 8$. The maximum error produced is 26%

Table 3.4. Crosstalk probability for 8-bit bus - SIG1 and SIG2

	SIG1			SIG2		
Bus line	Str-based	Stat.enum	Err(%)	Str-based	Stat.enum	Err (%)
1	0.50	0.50	0.0	0.50	0.50	0.0
2	0.62	0.61	1.6	0.62	0.62	0.0
3	0.62	0.61	1.6	0.62	0.63	1.6
4	0.61	0.62	1.6	0.62	0.61	1.6
5	0.62	0.61	1.6	0.62	0.58	6.8
6	0.61	0.57	7.0	0.63	0.50	26.0
7	0.62	0.63	1.5	0.62	0.55	12.7
8	0.32	0.32	0.0	0.11	0.11	0.0
		Average error: 1.9%				6.1%

Table 3.5. Crosstalk probability for 8-bit bus - SIG3 and SIG4

	SIG3			SIG4		
Bus line	Str-based	Stat.enum	Err(%)	Str-based	Stat.enum	Err (%)
1	0.51	0.51	0.0	0.49	0.49	0.0
2	0.62	0.60	3.2	0.63	0.61	3.1
3	0.62	0.60	3.2	0.62	0.60	3.2
4	0.62	0.60	3.2	0.63	0.61	3.1
5	0.60	0.60	0.0	0.61	0.61	0.0
6	0.62	0.60	3.2	0.56	0.60	7.1
7	0.58	0.60	3.4	0.56	0.60	7.1
8	0.46	0.46	0.0	0.34	0.34	0.0
		Average error: 2.0%				2.9%

while the average error is less than 7%. Table 3.6 compares the crosstalk probabilities for $m = 10$. The average error is less than 6%.

The run-times of the two proposed approaches are compared in Table 3.7. The statistical enumerative approach is shown to be considerably faster than the stream-based approach. This is because the statistical enumerator is independent of the length of the data stream while the execution time of the stream-based approach increases in direct proportion to the length of the data stream.

Table 3.6. Crosstalk probability for 10-bit bus - SIG1 and SIG2

SIG1				SIG2		
Bus line	Str-based	Stat.enum	Err(%)	Str-based	Stat.enum	Err (%)
1	0.48	0.49	2.0	0.49	0.50	2.0
2	0.61	0.63	3.2	0.61	0.62	1.6
3	0.61	0.62	1.6	0.61	0.62	1.6
4	0.61	0.62	1.6	0.62	0.62	0.0
5	0.61	0.62	1.6	0.61	0.61	0.0
6	0.61	0.62	1.6	0.61	0.62	1.6
7	0.60	0.62	3.2	0.61	0.61	0.0
8	0.61	0.59	3.4	0.62	0.56	10.7
9	0.60	0.59	1.7	0.61	0.57	7.0
10	0.38	0.43	11.6	0.43	0.33	30.3
		Average error: 3.2%				5.5%

Table 3.7. Stream-based estimator vs statistical enumerator run times

Data word size (bits)	Length	Stream-based (time)	Stat. enumerator (time)
8	450	14.45s	7.7s
8	3000	1min. 17s	7.2s
8	5000	2min. 13s	7.5s
8	6500	3min. 16s	7.8s
10	9500	4min. 22s	3min. 10s
10	14500	6min. 37s	3min. 11s

3.6 Conclusions

In this chapter, we presented a high-level, statistical enumerative technique for fast estimation of the crosstalk effects within a system bus, using the probability of crosstalk events as our metric. We compared it to a proposed stream-based technique that was shown to be faster than HSPICE, without any loss in accuracy. Although the speedup obtained in the statistical estimation process is significant, the algorithmic complexity of the estimator, with respect to the bus-width, is exponential. The following chapter addresses this issue.

CHAPTER 4

IMPROVING THE COMPLEXITY OF THE STATISTICAL ESTIMATION

The statistical enumerative crosstalk estimator discussed in the last chapter suffers from lack of scalability due to its high complexity with respect to the bus-width. In order to solve the complexity issue, we introduce some novel modifications to it. This chapter discusses a statistical non-enumerative technique that has linear time complexity with respect to the bus-width. We achieve the linear complexity by resorting to: (1) manipulation of the data stream to make the crosstalk-producing values continuous and (2) sampling the distribution function and storing it as a lookup table. Experimental results for data streams from different data environments are presented, compared against the stream-based approach. Average errors of less than 15% are obtained for bus-widths ranging from 8b to 32b. Further, due to the linearization of the complexity, the execution times are reduced by two orders of magnitude as compared to HSPICE.

4.1 Introduction and problem formulation

The modifications that are made to the statistical *enumerative* process to make it *non-enumerative* are stated as follows:

1. We use the *circular right shift* operation to change the position of the crosstalk templates in the concatenated data word. Such an operation has the effect of converting disjoint values in the enumerative process into continuous values. The summation of disjoint values in the enumerative process is reduced to a single definite integral.

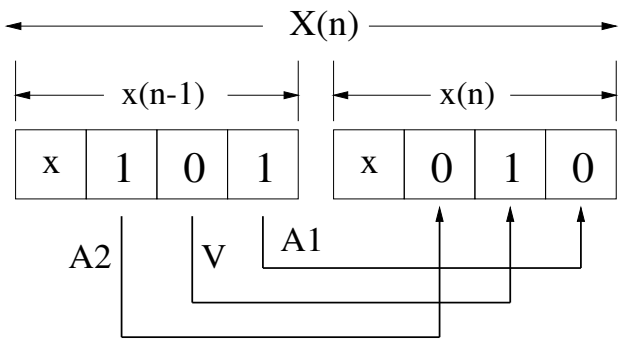


Figure 4.1. Concatenation

2. Further, we introduce a *sampling* technique to evaluate definite integrals for discrete-valued random variables. This linearizes the time complexity of the estimation process with respect to the bus-width.

4.2 Proposed statistical non-enumerative approach

As in the case of the statistical enumerative estimation process, we assume that the signal $x(n)$ on the bus has a normal distribution which is known *a priori*[11][12]. The signals are generated using ARMA equations. Besides, to avoid dealing with bit-level statistics, the temporal characteristics of the data stream are transformed to spatial characteristics, using the *concatenation* technique, as before. The word-level statistics of the concatenated data stream are derived analytically. The main equations pertaining to the probability distribution and concatenation of consecutive data words, are reproduced in equations 4.2 - **??**.

The probability p_i of the ith bit b_i in the data word on the bus is the probability that the ith line of the bus is a 1. This is given by:

$$p_i = Pr(x(n) \in \zeta_i)$$

$$= \sum_{j \in \zeta_i} \frac{1}{\sigma\sqrt{2\pi}} e^{-(j-\mu)^2/2\sigma^2} \tag{4.1}$$

where ζ_i is the set of all elements in ζ whose ith bit is 1.

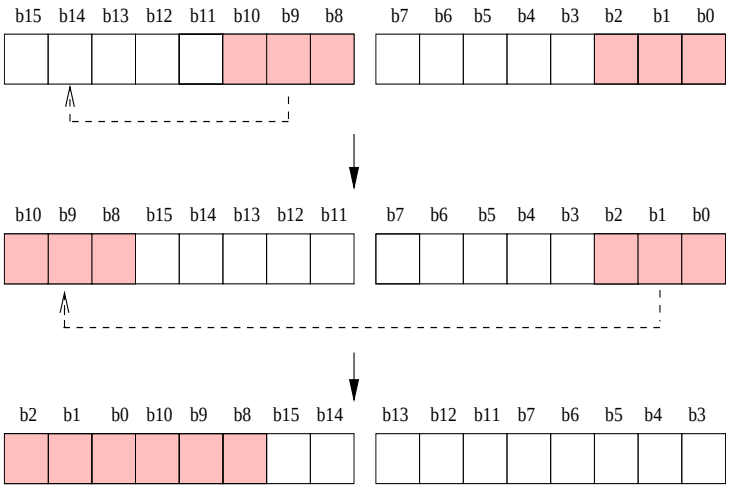


Figure 4.2. Continuous crosstalk windows using circular right shift

$$X(n) = 2^m x(n-1) + x(n) \tag{4.2}$$

$$\mu_X = (2^m + 1)\mu_x \tag{4.3}$$

$$\sigma_X = \sigma_x \sqrt{2^{2m} + 1 + 2^{m+1}\rho_x} \tag{4.4}$$

The concatenated data stream retains the nature of the probability distribution of the original input data stream to the bus. The statistics of the concatenated data stream $X(n)$ namely, mean μ_X and standard deviation σ_X, are utilized in the next step of the algorithm.

To scale the crosstalk estimation solution efficiently for larger bus-widths, the next step of the algorithm involves shifting the disjoint crosstalk windows to the MSB positions so that they are adjacent to one another, as shown in Figure 4.2. This is done using the *Circular Right Shift (CRS)* operation which obeys the following equation:

$$x'(n) = \lfloor x(n)/2 \rfloor + 2^{m-1}[x(n)\ mod\ 2] \tag{4.5}$$

where $x'(n)$ is obtained by shifting $x(n)$ once to the right in a circular fashion. From equation 4.5, we observe that the *CRS* operation also preserves the normal properties of the distribution because of its linear nature.

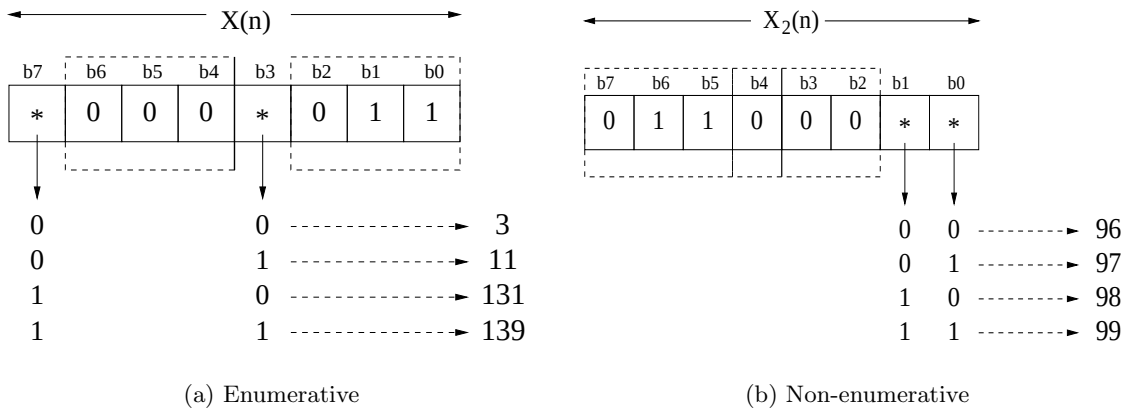
Each *CRS* operation causes the statistics of a data stream to change. Using the above *CRS* equation, we derive analytical equations that relate the mean and standard deviation of the shifted data stream $x'(n)$ to those of the original stream $x(n)$ as shown:

$$\begin{aligned} E[x'(n)] &= E\lfloor x(n)/2 \rfloor + 2^{m-1} E[x(n)\ mod\ 2] \\ \mu_{x'} &= \frac{\mu_x}{2} + 2^{m-1}\eta \end{aligned} \tag{4.6}$$

$$\begin{aligned} \sigma^2_{x'} &= E[x'^2] - \mu^2_{x'} \\ &= E[\frac{x(n)^2}{4} + 2^{2(m-1)}[x(n)\ mod\ 2]^2 \\ &\quad +2^{m-1}x(n)[x(n)\ mod\ 2]] \\ &\quad -\frac{\mu_x^2}{4} - 2^{m-1}\eta\mu_x - 2^{2(m-1)}\eta^2 \\ &\quad -\frac{\mu_x^2}{4} - 2^{m-1}\eta\mu_x - 2^{2(m-1)}\eta^2 \\ &= \frac{E[x(n)^2]}{4} + 2^{2(m-1)}\eta + 2^{m-1}\mu_x\eta - \frac{\mu_x^2}{4} \\ &\quad -2^{m-1}\eta\mu_x - 2^{2(m-1)}\eta^2 \\ &= \frac{\sigma_x^2}{4} + 2^{2(m-1)}(\eta - \eta^2) \end{aligned} \tag{4.7}$$

where $\eta = E[x(n)\ mod\ 2]$ is the bit probability of the current LSB in $x(n)$.

This CRS operation is performed in two stages to create continuous crosstalk windows. It is first applied to the left half of $X(n)$ to get a modified $2m$-bit value $X_1(n)$ Now, the CRS operation is again performed on $X_1(n)$ to get the transformed value $X_2(n)$. This makes all the bits in a given crosstalk pattern adjacent to each other and located in the *MSB* region



(a) Enumerative (b) Non-enumerative

Figure 4.3. Enumerative & non-enumerative techniques for a 4-bit bus with template instance 000011 and victim b1

of $X_2(n)$. Consequently, dispersed values in the original $X(n)$ map to continuous values in $X_2(n)$.

Consider a 4-bit bus as an example. The compound word $X(n)$ is 8-bits (*b7-b0*) wide. Now, if bits *b6-b4* = '000' and bits *b2-b0* = '011', it corresponds to a transition that causes crosstalk on *b1*. By substituting '00' for bits *b7 b3*, we obtain the value 3 which represents a crosstalk producing transition for *b1*. Using the CRS technique, we now shift the crosstalk pattern to the MSB position of the 8-bit bus. The pattern now reads '011000'. By substituting '00' for bits *b1 b0*, we obtain the value 96. Thus, the disjoint values {3, 11, 131, 139} in $X(n)$ map to the continuous values {96, 97, 98, 99} in $X_2(n)$. As shown in Figure 4.4, exhaustive enumeration of values *v1, v2, v3*, and *v4* reduces to the integral between limits *l1* and *l2* with $l1 = 96$ and $l2 = 99$. Figure 4.3 illustrates the examples.

This modification reduces the complexity of the algorithm from exponential (in the disjoint case) to linear (in the continuous case) with respect to the bus-width m.

4.2.1 Evaluation of the definite integral using sampling

For large bus-widths, the bounds of the definite integrals are far apart. Since the word-level values are discrete in nature, we propose a *sampling* technique to evaluate the integrals in such cases. This provides a fast and accurate solution.

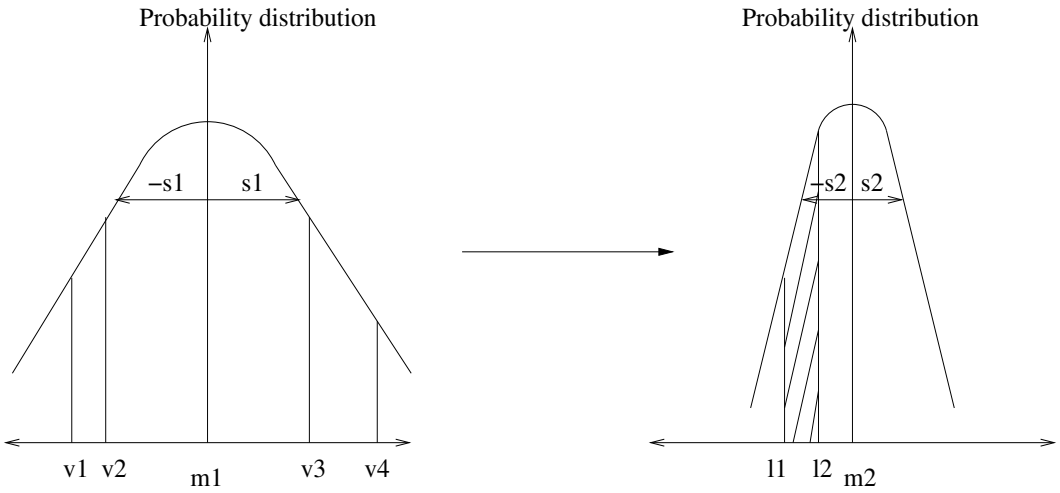


Figure 4.4. Enumeration to integral transformation

Each interval corresponding to an integral is sampled a specific number of times. Each value obtained by substituting the sample into the probability distribution function is stored in a *look-up table*. The interval is now split up into sub-intervals whose width is given by:

$$w_{si} = \frac{ub - lb + 1}{n} \tag{4.8}$$

where w_{si} is the width of each sub-interval and the integral is bounded by [*lb, ub*] and sampled n times. This is illustrated in Figure 4.5. Thus, each sub-interval is bounded by values which are stored in the look-up table. We evaluate the integral for each sub-interval by taking the product of the median for the sub-interval and width w_{si}. If the values stored in the look-up table are v_1, v_2, v_3, ..., v_n, the integral I_j for the j*th* sub-interval is given by:

$$I_j = \frac{v_j + v_{j+1}}{2} * w_{si} \tag{4.9}$$

The integrals for the remaining sub-intervals are also evaluated in a similar manner. We then compute the integral I_{int} for the interval [*lb, ub*] by summing up the integrals for all the sub-intervals. Thus,

$$I_{int} = \sum_{j=1}^{n-1} I_j \tag{4.10}$$

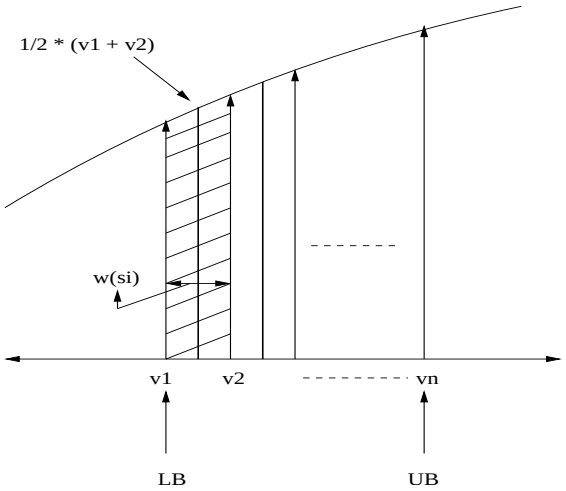


Figure 4.5. The sampling technique

Table 4.1. Data environments

Bit-width	Data env	ARMA equation
8	SIG 1	x(n)=75γ(n)+200
16	SIG 2	x(n)=250γ(n)+56*10^3
32	SIG 3	x(n)=$10^6\gamma$(n)+0.5x(n-1)+5*10^8

For each of the middle lines, the crosstalk probability is computed by summing up such integrals. For the edge lines, the crosstalk probability is obtained directly from the transition activity of their aggressors, using Equation (3.11). The non-enumerative procedure flow is illustrated in Figure 4.6.

4.3 Experimental results

We compare the proposed non-enumerative statistical crosstalk estimation technique with the stream-based technique in terms of both accuracy and speed for different data environments.

Table 4.1 shows the data environments modeled using ARMA models as before. The proposed approach is general enough to handle any data environment. This means that it will work on all data streams from such an environment. The white noise factor γ(n) has a standard normal distribution.

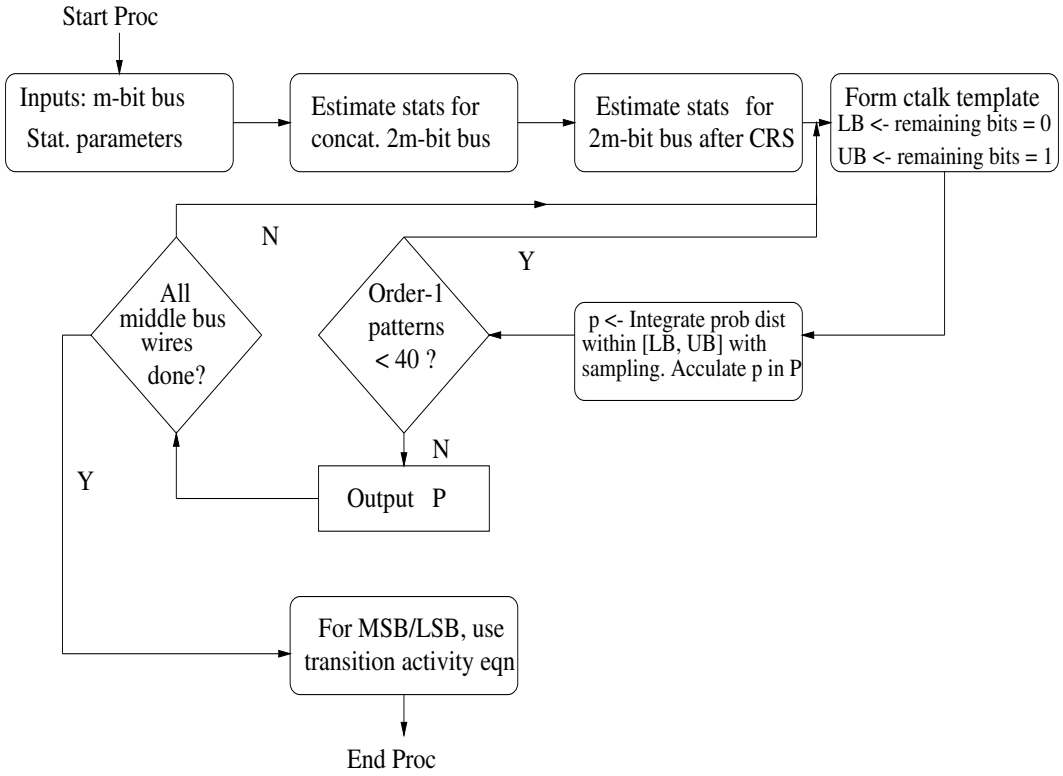


Figure 4.6. Non-enumerative statistical crosstalk probability estimation flow

Table 4.2. Crosstalk probability for 8-bit bus

Bus line	Stream-based	Stat.estimator	Err(%)
1	0.51	0.51	0.0
2	0.62	0.58	6.4
3	0.63	0.59	6.3
4	0.65	0.58	10.7
5	0.62	0.57	8.0
6	0.63	0.57	9.5
7	0.51	0.69	35.2
8	0.49	0.49	0.0

Average error: 9.5%

Table 4.3. Crosstalk probability for 8-bit bus - SIG1 & real audio data

SIG 1			Real audio data	
Bus line	Str-based	NET	Str-based	NET
1	0.51	0.51	0.43	0.41
2	0.62	0.58	0.52	0.50
3	0.63	0.59	0.55	0.50
4	0.65	0.58	0.48	0.44
5	0.62	0.57	0.43	0.43
6	0.63	0.57	0.44	0.43
7	0.51	0.69	0.50	0.58
8	0.49	0.49	0.32	0.37

Tables 4.2 - 4.5 compare the crosstalk probabilities as computed by the statistical approach against those obtained from the stream-based estimator for bus-widths ranging from 8 bits to 32 bits. The average error for the entire bus for m=8b is 9.5% while for m=16b and m=32b, the average errors are 5.9% and 14.9% respectively.

Besides the ARMA models, we test the proposed approach using real audio data from a human voice. The audio was recorded using the *audiotool* utility in UNIX. The results are shown in Table 4.3. The average error is 7.4%.

It may be noted that although the probability error in line 25 for the 32-bit bus is very high, it differs only in the second decimal place. In practice, it has only 9% chance of crosstalk which we estimate to be negligible.

The runtimes for the proposed non-enumerative technique are compared to those of the stream-based estimator for different bus-widths in Table 4.6. It is to be noted that with increase in the data stream length, the difference in the runtimes becomes even more significant.

Increasing the number of samples increases the accuracy as well as the runtimes. It is observed that beyond a certain number of samples, the increase in accuracy is insignificant as compared to the run times. Hence, the number of samples to be used for each interval should be discreetly selected. Table 4.7 shows the effects of the number of samples on the runtimes.

Table 4.4. Crosstalk probability for 16-bit bus

Bus line	Stream-based	Stat.estimator	Err(%)
1	0.53	0.53	0.0
2	0.61	0.70	14.7
3	0.66	0.70	6.1
4	0.63	0.69	9.5
5	0.63	0.69	9.5
6	0.62	0.68	9.7
7	0.62	0.68	9.7
8	0.65	0.68	4.6
9	0.60	0.75	25.0
10	0.51	0.54	5.9
11	0.46	0.46	0.0
12	0.15	0.15	0.0
13	0.00	0.00	0.0
14	0.00	0.00	0.0
15	0.00	0.00	0.0
16	0.00	0.00	0.0

Average error: 5.9%

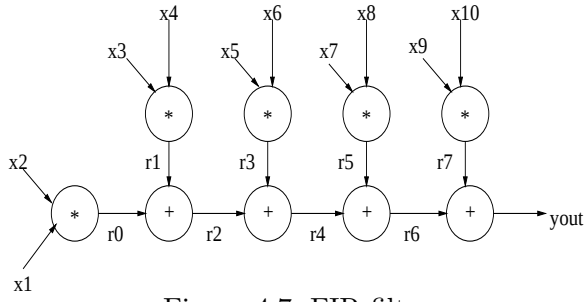


Figure 4.7. FIR filter

Table 4.5. Crosstalk probability for 32-bit bus

Bus line	Stream-based	Stat.estimator	Err(%)
1	0.31	0.27	12.9
2	0.50	0.52	4.0
3	0.59	0.46	22.0
4	0.65	0.48	26.2
5	0.62	0.49	20.9
6	0.61	0.49	19.7
7	0.64	0.50	21.8
8	0.64	0.50	21.8
9	0.64	0.50	21.8
10	0.61	0.50	18.0
11	0.62	0.47	24.2
12	0.62	0.49	20.9
13	0.61	0.49	19.7
14	0.62	0.50	19.3
15	0.59	0.50	15.2
16	0.63	0.50	20.6
17	0.63	0.50	20.6
18	0.64	0.50	21.9
19	0.60	0.50	16.7
20	0.64	0.50	21.9
21	0.57	0.50	12.3
22	0.53	0.44	17.0
23	0.35	0.47	34.3
24	0.13	0.16	23.0
25	0.09	0.01	88.9
26	0.00	0.00	0.0
27	0.00	0.00	0.0
28	0.00	0.00	0.0
29	0.00	0.00	0.0
30	0.00	0.00	0.0
31	0.00	0.00	0.0
32	0.00	0.00	0.0

Average error: 14.9%

Table 4.6. Stream-based estimator vs statistical estimator run times

Bus-width	Length	Stream-based	Stat estimator
8b	1000	22.4s	0.4s
16b	1000	24.3s	1.2s
32b	1000	33.0s	0.5s

Table 4.7. Effect of samples(s) on runtimes

Bus-width	s=1000	s=5000	s=50000
16	1.2s	2.6s	10.0s
32	0.5s	1.7s	15.0s

We demonstrate the application of the non-enumerative statistical estimator to compute the crosstalk probabilities of each line on each edge of a finite-impulse response (FIR) filter. The data-flow graph for the FIR filter is shown in Figure 4.7. Each edge of the filter is an 8-bit bus and the proposed estimator is run using the word-level statistics on that edge. The statistics themselves are propagated using the technique proposed by Ramprasad et.al. [11]. The crosstalk estimate for each line of an edge is compared against the stream-based program. For simplicity, only the average errors are reported for each edge in Table 4.8. The word-level statistics for the primary inputs are generated using the ARMA model SIG 1 from Table 4.1.

Finally, in order to demonstrate the compatibility of our approach with an existing crosstalk minimization technique, namely, bus re-ordering, we re-order the bus lines by placing two of the lines with the lowest crosstalk susceptibility between bus lines with the highest susceptibility. Table 4.9 shows the decrease in crosstalk probabilities, following the re-ordering. Table 4.10 gives the crosstalk probabilities obtained for the bus lines using the non-enumerative statistical estimation procedure for both the original as well as the re-ordered bus, compared against the probabilities obtained using the stream-based verification procedure. The average estimation error is seen to decrease slightly from 3.7% for the original bus ($\mu = 109.02, \sigma = 83.2, \rho = 0.8$) to 2.9% for the re-ordered bus ($\mu = 100.96, \sigma = 69.2, \rho = 0.4$). This study demonstrates that the proposed statistical non-enumerative technique is reliable.

4.4 Conclusions

We presented a non-enumerative statistical technique to evaluate bit-level probability of crosstalk events within a system bus from word-level statistical parameters of the input

Table 4.8. Avg. crosstalk estimation error - FIR

Edge_id	Avg. err(%)	Edge_id	Avg. err(%)
x1	8.7	yout	15.6
x2	8.7	r0	8.7
x3	9.0	r1	11.4
x4	8.8	r2	15.5
x5	9.3	r3	11.6
x6	8.7	r4	21.0
x7	9.3	r5	9.2
x8	9.0	r6	30.4
x9	8.7	r7	12.0
x10	9.4	-	-

Table 4.9. Crosstalk probabilities on original and re-ordered bus lines and decrease in crosstalk susceptibility due to re-ordering

Bus line	Original	Re-ordered	Decrease in crosstalk(%)
1	0.30	0.20	+33.3
2	0.62	0.63	-1.6
3	0.64	0.61	+4.7
4	0.61	0.56	+8.2
5	0.61	0.64	-4.9
6	0.57	0.53	+7.0
7	0.54	0.56	-3.7
8	0.20	0.20	0.0

Table 4.10. Estimation probabilities for original and re-ordered bus

	Original bus			Re-ordered bus		
Bus line	Str.based	NET	Err(%)	Str.based	NET	Err(%)
1	0.30	0.30	0.0	0.20	0.20	0.0
2	0.62	0.59	4.8	0.63	0.60	4.7
3	0.64	0.60	6.3	0.61	0.60	1.6
4	0.61	0.59	3.3	0.60	0.60	0.0
5	0.61	0.57	6.6	0.64	0.60	6.2
6	0.57	0.52	8.7	0.53	0.57	7.5
7	0.54	0.54	0.0	0.56	0.54	3.5
8	0.20	0.20	0.0	0.20	0.20	0.0

data. We introduced a sampling technique to quickly evaluate definite integrals of discrete random variables during the estimation process. The technique reduces the estimation complexity from exponential to linear with respect to the bus-width. The technique has been efficiently applied to estimate *intra-bus crosstalk*. The following chapter addresses the *inter-bus* crosstalk estimation problem at the layout level of designs. The statistical technique for intra-bus crosstalk estimation is integrated along with a floorplanner and a global router to predict inter-bus crosstalk effects.

CHAPTER 5

FLOORPLAN-BASED CROSSTALK ESTIMATION FOR MACROCELL BASED DESIGNS

In this chapter, we address the critical problem of crosstalk estimation between the different buses at the layout level of a design. This problem is significant in that it's solution provides crosstalk susceptibility estimates of victim wires that enable designers to optimize a design for crosstalk. We propose an estimation technique to measure the crosstalk susceptibility of different nets in the post global routing phase, prior to detailed routing of designs. Global routing provides the approximate routes of the wires. This is used to compute the aggressors of a given victim wire along its route and its crosstalk susceptibility with respect to those aggressors. The crosstalk susceptibility of a wire is given by: (1) P_t, the probability of crosstalk occurrence on the wire in different regions along its route; and (2) V_{peak}, the worst case crosstalk noise amplitude experienced by the wire along its route. P_t is estimated using a very fast and accurate statistical estimator previously proposed by the authors. V_{peak} is estimated by predicting the cross-coupling capacitances between neighboring wires, using their global routing information. Placement and global routing are done using CADENCE Silicon Ensemble. The predicted crosstalk estimates are compared against those by detailed HSPICE simulations. Average errors are found to be less than 8%.

5.1 The significance of the inter-bus crosstalk problem

The inter-bus crosstalk estimation problem is a precursor to generating crosstalk-immune designs. Accurate estimation of the crosstalk susceptibility of victim nets at the layout-level allows the designer to repeat the physical-level synthesis steps with crosstalk minimization

as the objective function. This process may be iterative in nature - starting with an initial layout, the designer may estimate the crosstalk susceptibility of the victim nets and may feed this information back to the placement or routing phases of the synthesis process. The router may then re-route the victims which are most susceptible and generate a new layout. The crosstalk susceptibility of the victims in the new layout is then reduced without violating other timing constraints.

In this chapter, we propose a technique to estimate the inter-bus crosstalk which uses the statistical intra-bus crosstalk estimation process that we proposed and described in the last two chapters. Given only word-level statistics on a bus, the intra-bus crosstalk estimator computes a bit-level probability for each line of the bus. Our intra-bus crosstalk estimator was shown to have good accuracy with upto two orders of magnitude in speedup. We choose to leverage on this quick and accurate technique with a view to establishing the dependence of crosstalk at the layout-level, on the input statistics on the various wires. Previously, the dependence of dynamic power consumption on the input statistics to a circuit was established in [77][78][79][80].

However, the intra-bus crosstalk estimation was at the behavioral-level of a design and assumed all the lines of a bus to be placed next to one another for the entire length of the bus. This assumption, while reasonable for a system bus running from a fixed source to a destination, may not be true during physical synthesis. The routing is typically performed wire-by-wire rather than for individual buses. Thus, it is possible for wires with common sources and destinations to diverge at some point in the circuit and re-converge at a later point in the circuit. Moreover, the inter-bus crosstalk estimation demands that certain physical-level parameters of the design are accurately accounted for. Hence, we adopt an approach which integrates our intra-bus estimator with the physical-level information available to us at the post-global routing phase of the design.

There have been previous attempts to solve the crosstalk estimation and reduction problem in the post-global routing phase of synthesis [81][66][82][83]. These techniques are targeted towards PCBs as well as Multi-chip modules (MCMs). Some of the routing

techniques used for crosstalk minimization are also strong enough to handle yield improvements, by minimizing the chances of open circuits and short circuits [84]. However, these techniques rely on runtime-intensive graph manipulation techniques for crosstalk estimation and area-intensive shielding techniques for crosstalk reduction. While these are constructive techniques which take a crosstalk cost function into account while solving the routing problem, other techniques such as [85][86] are iterative in nature. Delay as well as crosstalk spikes are minimized by re-adjusting the space between the interconnect lines of a routed design. Alternatively, successive *ripup and rerouting* of individual wires leads to increasingly better solutions. Crosstalk inside switchboxes during routing has been studied in [87] and ILP-based solutions have been proposed for their minimization. These solutions, although novel, are proposed at the physical-level of design abstraction. They need precise physical level information about the circuit in order to run. On the other hand, our approach is to gather only a small fraction of the physical level information and predict the rest so that we can leverage on the statistical estimation techniques that we previously proposed for the intra-bus crosstalk estimation. Thus, we are able to take advantage of the fast design-space exploration at the behavioral and RT-levels while estimating effects at the physical-level.

In order to generalize our technique for a circuit layout, in our current work, we form *composite buses* based on global route information about neighboring wires provided by the floorplanner. Besides, we extend the approach to use the same physical-level information to compute a worst-case crosstalk noise pulse for each wire in the design.

The organization of this chapter is as follows: Section 5.2 describes the features of the place and route tool used in this work. Section 5.3 details how we decompose the inter-bus crosstalk problem so that it transforms to the model we used for the intra-bus crosstalk problem. Section 5.3, subsection 5.3.1 validates the empirical predictions we make in the post-global routing phase of the design with respect to the physical orderings between victim wires in different routing regions at the layout level. This information is critical in estimating the cross-coupling capacitance between the victim wires. Section 5.3, subsection 5.3.2 also describes how we obtain the statistics on the aggressors of a victim. Section 5.3,

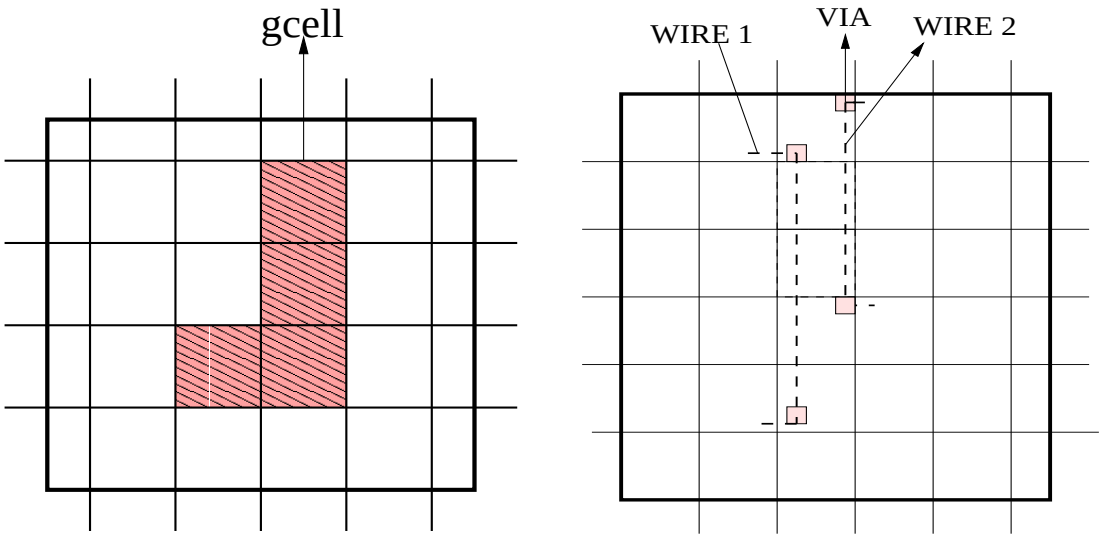
subsection 5.3.3 forms *composite buses* to get the word-level statistics along all the routing regions of a victim wire. Section 5.4 details the analytical evaluation of the two crosstalk susceptibility metrics namely, the crosstalk probability on a victim wire and its worst-case crosstalk noise amplitude along different regions during its route. This section also combines the individual crosstalk probability and worst-case noise amplitude values along the victim's route into a single value for the crosstalk probability and a single value for the crosstalk noise amplitude for the entire victim wire. Section 5.5 presents experimental results and their analysis. Finally, Section 5.6 draws conclusions.

5.2 The place and route tool

To accurately predict crosstalk effects between different wires in the layout, we utilize floorplan information about the design. This information is obtained from Silicon Ensemble, a commercial physical synthesis tool from Cadence. For macrocell-based designs, Silicon Ensemble superimposes a *grid* on top of a generated placement. Each cell in the grid is known as a *global routing cell* (gcell). Every wire passes through a sequence of gcells while traversing from the source terminal to the destination terminal, as shown in Figure 5.1.

Separate metal layers are used for the horizontal and vertical route of every wire. Thus, the approximate route of any wire in the design is the sequence of gcells through which it passes either horizontally or vertically. Consequently, the wires which pass through the same gcell in the horizontal direction are in the vicinity of one another and are considered neighbors. Similarly, the wires which pass through the same gcell in the vertical direction are also neighbors of one another. For example, in Figure 5.1(b), *WIRE 1* and *WIRE 2* which share two gcells in the vertical direction, are neighbors in both the cells. Vias are used at all intersections to transfer from the horizontal layer to the vertical layer and vice versa.

The global routing phase does not give the ordering of the wires within a gcell. It merely gives the approximate route of each wire. The ordering of the wires within any gcell is predicted using the placement information. For the horizontal adjacency, it is assumed,



(a) Grid-based routing

(b) Neighboring wires on the grid

Figure 5.1. Global routing of wires

based on empirical observations, that if *module 1* is placed above *module 2*, then a wire originating from module 1 is also placed above a wire originating from module 2. For the vertical adjacency, it is similarly assumed that if *module 1* is placed to the left of *module 2*, then a wire originating from module 1 is to the left of the wire originating from module 2.

Thus, after placement and global routing phase of the tool we obtain the following information:

- The approximate route of each net.
- The neighboring wires inside a gcell.
- The *ordering* of the neighbors within the gcell.

Having obtained the above, we formulate the inter-wire crosstalk estimation problem so that it can be efficiently incorporated into the previously proposed statistical estimation technique. The procedure is split into several steps. The flow is shown in Figure 5.4. The different steps are explained subsequently.

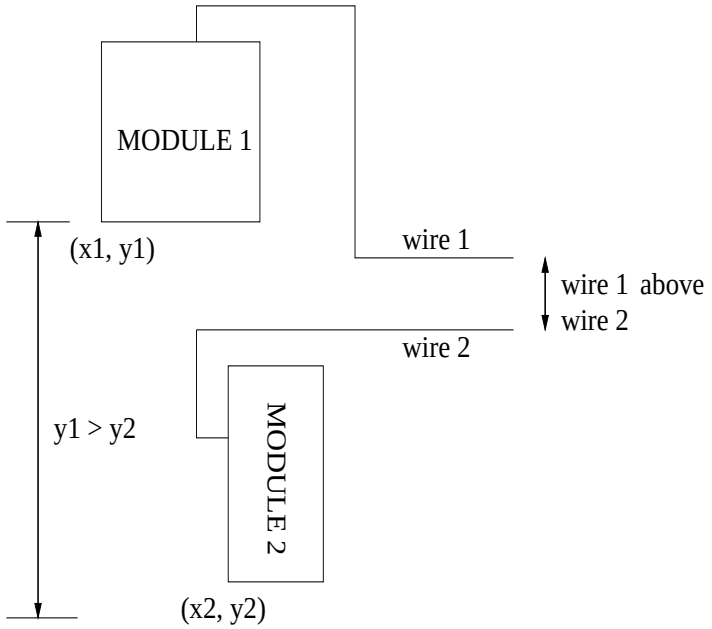


Figure 5.2. Horizontal ordering of wires during global routing

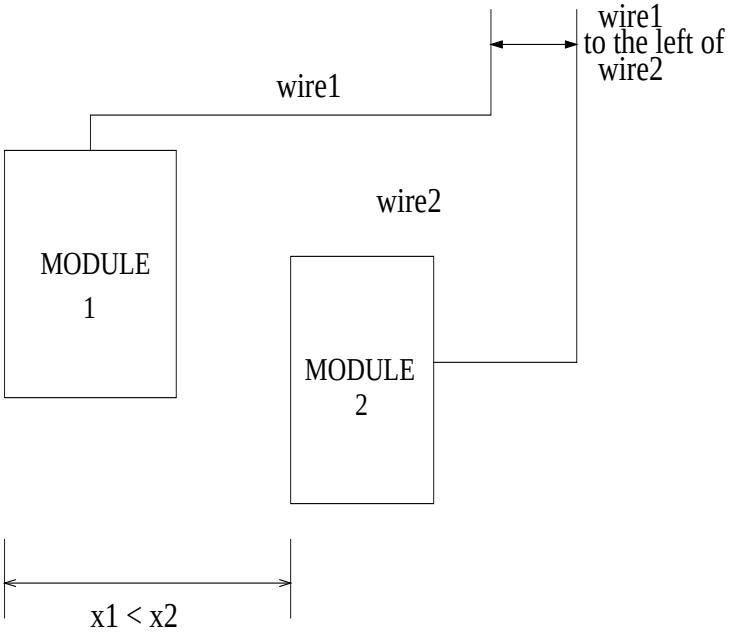


Figure 5.3. Verical ordering of wires during global routing

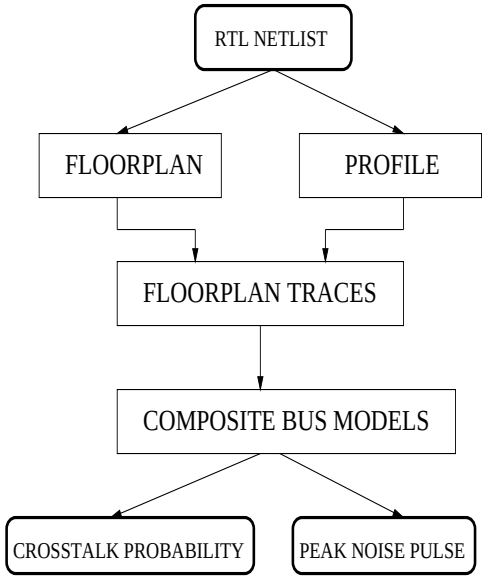


Figure 5.4. General procedure flow

5.3 Modeling the inter-wire crosstalk problem

This section explains in detail how we model the inter-wire crosstalk estimation problem so that it fits into the intra-bus crosstalk estimation model that we previously used. Modeling the inter-wire crosstalk estimation problem involves the following steps:

- Establishing the wire-ordering assumptions.
- Profiling the design at the RT-level so as to obtain the bit values on each wire.
- Forming the composite buses within each global routing cell and estimating the coupling capacitances between the wires of each bus.

The following subsections describe each of these steps in detail.

5.3.1 Validating the empirical wire-ordering assumptions

The global route-level assumptions regarding the exact wire ordering at the detailed routing level may be formally stated as follows:

Table 5.1. Wire-ordering validation

Benchmark	Horizontal	Vertical
DiffEq	70.1%	66.8%
FIR	74.3%	59.0%
IIR	55.3%	71.0%
FFT4pt	65.3%	69.2%
DCT2pt	67.0%	71.1%

- If *wire 1* originates from a module whose placement coordinates are $(x1, y1)$ and *wire 2* originates from a module with placement coordinates $(x2, y2)$, then $y1 > y2 \Longrightarrow$ *wire 1* is placed above *wire 2*.

- If *wire 1* originates from a module whose placement coordinates are $(x1, y1)$ and *wire 2* originates from a module with placement coordinates $(x2, y2)$, then $x1 < x2 \Longrightarrow$ *wire 1* is placed to the left of *wire 2*.

Figures 5.2 and 5.3 illustrate the ordering assumptions. In order to determine the correctness of our empirical assumptions, we compare them with the actual wire orderings obtained after the detailed routing phase. Table 5.1 validates the accuracy of the assumptions for both horizontal and vertical adjacencies over the various benchmarks.

5.3.2 RT-level profiling

The first step involves obtaining the register-transfer level (RTL) netlist from a behavioral description of a design and computing data values on the edges of the RTL design. For the process of high-level synthesis, we use a high-level synthesis system developed by the authors. It inputs a behavioral description of the design in the form of a data flow graph. It then performs the high-level synthesis steps namely, scheduling, allocation, and binding to generate the corresponding register-transfer level design in the form of structural VHDL, using a library of pre-defined components. The simulation dump of the VHDL design provides us with the data bits on the input, intermediate, and output edges of the RTL netlist.

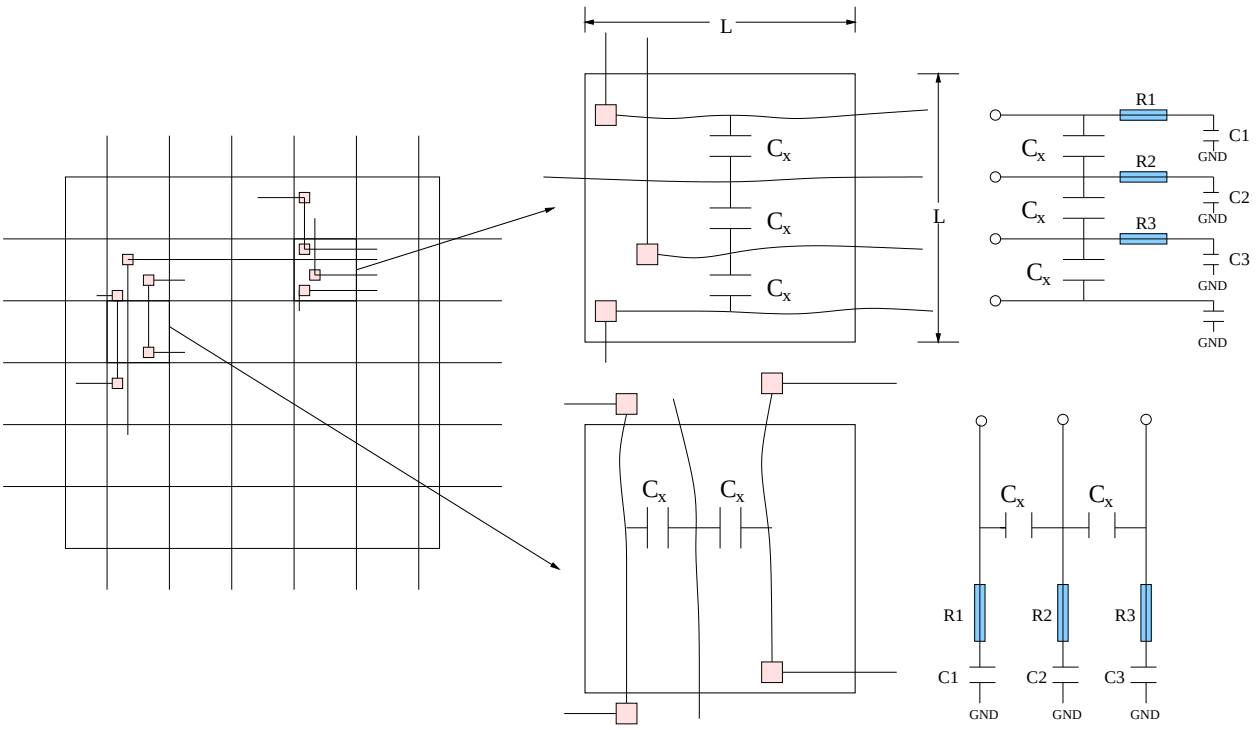


Figure 5.5. Formation of the composite bus from global route

5.3.3 Composite bus formation

The notion of different wires acting as neighbors inside a global routing cell enables us to define a *composite bus* for every gcell. Formally, a composite bus within a gcell is defined as a set of wires with same or different sources and destinations which pass through this gcell in the same direction (horizontal or vertical) along their route. It is necessary for the wires to pass the gcell in the same direction (horizontal or vertical). Only then will it be ensured that the wires are coplanar and will give rise to crosstalk effects. Crosstalk effects between perpendicular wires in different metal layers are considered to be negligible in this work since the overlap length is very small.

Each gcell has a list of different wires passing through it. Together, they constitute the *composite* bus. It should be noted that the completion of the global routing phase does not fix up the exact coordinates of a route. Hence, the wires are still *flexible* with respect to their positions. Their exact coordinates will be determined only after detailed routing.

Therefore, in the post global routing phase, we use a *prediction* method to estimate the separation between the wires and thereby, the cross-coupling capacitance C_x. The details of the prediction method are explained subsequently.

Figure 5.5 illustrates the formation of the composite bus. Two global routing cells are shown along with the wires passing through them in the horizontal and vertical directions. The magnified figures of the gcells shows the flexible, coplanar wires with cross-coupling capacitances between themselves. Besides, each wire has an associated wire-to-substrate capacitance as well as a resistance. Thus, the composite bus is converted to the traditional crosstalk model for bus-based interconnects.

5.4 Estimating the inter-wire crosstalk metrics

This section describes the estimation of crosstalk susceptibility of each net. The crosstalk susceptibility computation involves (i) crosstalk probability estimation for each wire and (ii) the maximum crosstalk noise pulse amplitude estimation for each wire. The following subsections describe the estimation of each of these parameters in detail.

5.4.1 Crosstalk probability estimation

The RT-Level profiling gives us the data values on all the wires of the RTL design. Since the placement phase involves the cells constituting the RTL datapath and controller while the global routing phase involves the wires in the RTL interconnect, the data values on the wires are known in the post placement and global routing phase. In other words, for a given gcell, the data transitions on every wire of a composite bus are known. Hence, we are able to compute the statistical parameters namely, mean, standard deviation, and lag-1 temporal correlation for the composite bus. The statistical estimator, which takes these parameters as inputs, quickly evaluates the crosstalk probability of every wire in the global routing cell under consideration. The statistical estimator is run once for every gcell along a wire's route and for all the wires in the synthesized design.

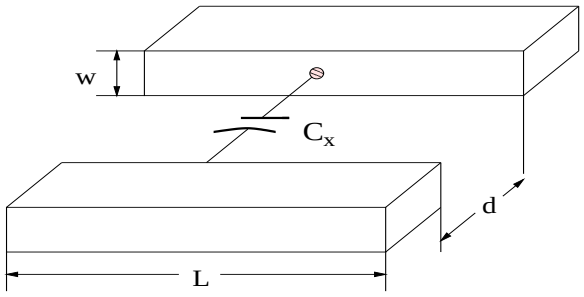


Figure 5.6. Cross-coupling in parallel wires

For a given wire, we are thus able to compute a series of crosstalk probabilities, corresponding to every gcell through which the wire is routed. These individual probabilities are compared against detailed HSPICE simulation of RC models corresponding to those same gcells. The formation of the RC models is explained in the experimental results section.

5.4.2 Maximum noise pulse estimation

The maximum noise pulse on a victim wire within a global routing cell is given in [16]. For the convenience of the reader, it is reproduced in Equation 5.1.

$$V_P = \frac{1}{1 + \frac{C_2}{C_X} + \frac{R_1}{R_2}(1 + \frac{C_1}{C_X})} \tag{5.1}$$

where C_X is the cross-coupling capacitance between the aggressor and victim wires. C_1 and C_2 are the wire-to-ground capacitances while R_1 and R_2 are the lumped resistances of the aggressor and victim wires respectively. The parameters C_1, C_2, R_1, and R_2 are fixed for a given target technology. On the other hand, the cross-coupling capacitance C_X not only depends on the length of overlap between the aggressor and victim but also on their separation. Thus, this parameter will vary from gcell to gcell.

The aggressor and victim wires obey the equation for the parallel plate capacitor given below:

$$C_X = \frac{\epsilon_o \kappa A}{d} = \frac{\epsilon_o \kappa L * w}{d} \tag{5.2}$$

where ϵ_o and κ are the permittivity of air and the dielectric constant respectively, L is the overlap length between the wires, w is the width of the overlapping areas of the wires, and d is the separation distance, as shown in Figure 5.6. Thus, for a fixed overlapping area and dielectric,

$$C_X \propto \frac{1}{d} \Longrightarrow \frac{C_{X,new}}{C_{X,org}} = \frac{d_{org}}{d_{new}} \tag{5.3}$$

The dimensions LxL of the gcells are provided by the technology file used. If there are m wires which constitute a composite bus within a gcell, then the inter-wire separation d_{sep} is estimated as

$$d_{sep} = \frac{L}{m} \tag{5.4}$$

The original inter-wire separation distance d_{org} is assumed to be 0.6μm which is the track separation in the layouts. The original cross-coupling capacitance $C_{X,org}$ is computed with this wire separation and Equation 5.2. Thus, within every gcell, the cross-coupling capacitance between the wires of the composite bus is given by

$$C_{X,gcell} = C_{org} * \frac{d_{org}}{d_{sep}} \tag{5.5}$$

$$= C_{org} * \frac{m d_{org}}{L} \tag{5.6}$$

Once the cross-coupling capacitances are computed, the maximum noise pulse of a victim net within a gcell is analytically estimated using Equation 5.1. The estimated noise pulse is compared against that from the detailed HSPICE simulation of the composite bus wires within the gcell.

5.4.3 Modification of the estimation process using uniform-wire models - rms estimate

Although the above technique of crosstalk probability estimation is accurate with respect to the individual global routing cells, it still suffers from the drawback of not formulating a single crosstalk probability measure and a single amplitude estimate for an entire victim net. Thus, the effect of the same neighbor of a victim which acts as its aggressor in two different gcells will be evaluated twice, once for each gcell. Since there is a complete overlap in the crosstalk events which will occur on the victim as a result of this aggressor, there will be a redundancy in the estimation process. In order to alleviate this drawback, we propose a cost function that incorporates the crosstalk probability of a victim net inside each global routing cell along its route to compute a single crosstalk probability for the entire victim net which accounts for the event overlaps.

The cost function used is the *root mean square* (rms) value of the crosstalk probabilities from different gcells along a victim wire's route. By definition, the rms value of a varying quantity is a statistical measure of its magnitude. Thus, once we obtain the crosstalk probability within each gcell along the route of a specified victim wire, the resultant crosstalk probability of the wire is given by

$$Pr_{wire} = \sqrt{\frac{\Sigma p_{gcell}^2}{n}} \tag{5.7}$$

where Pr_{wire} is the resultant crosstalk probability of the entire wire, p_{gcell} is the probability of the wire within the i^{th} gcell, and there is a total of n gcells along the entire route of the wire. We demonstrate that the RMS value of crosstalk probabilities takes the overlapping events from the same aggressors in different gcells into account.

Lemma: Given k global routing cells along the path of a victim wire with the same aggressor, the overlapping crosstalk events on the victim can be captured using the root mean square value of the crosstalk probabilities in each of the k gcells.

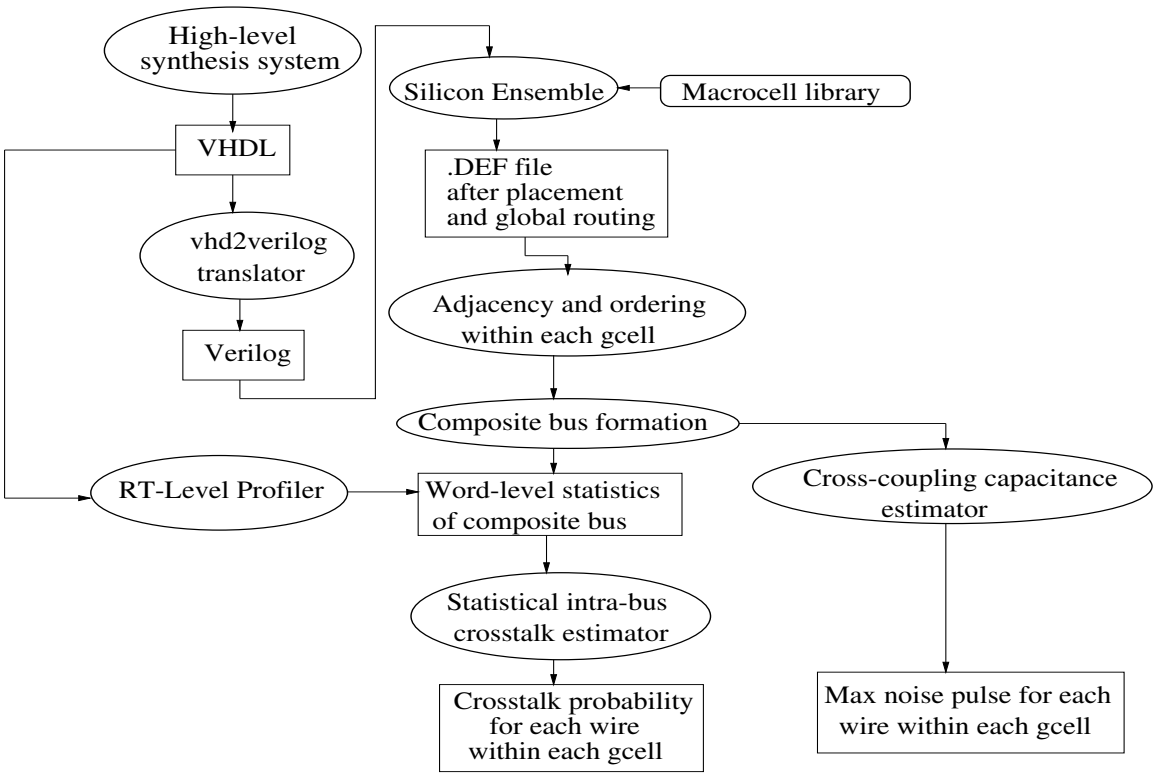


Figure 5.7. Experimental flow

Proof: Since there are an equal number of crosstalk events in each of the k gcells, the crosstalk probabilities in these cells are identical. Let this crosstalk probability be p_{gc}. Then, the RMS value for k gcells is computed as

$$\begin{aligned} p_{rms} &= \sqrt{\frac{\Sigma p_{gc}^2 + p_{gc}^2 + p_{gc}^2 + ...ktimes}{k}} \\ &= \sqrt{\frac{k * p_{gc}^2}{k}} \\ &= \sqrt{p_{gc}^2} = p_{gc} \end{aligned} \tag{5.8}$$

This is the same crosstalk probability as that of the overlapped crosstalk events. Moreover, $0 \leq p_{gc} \leq 1$ implies that $0 \leq p_{rms} \leq 1$

5.5 Experimental results

The detailed experimental flow is given in Figure 5.7. The output of the high-level synthesis system namely, the RT-Level netlist of a design is converted from structural VHDL to verilog and fed into Cadence Silicon Ensemble. The other input to Silicon Ensemble is the pre-characterized macrocell library, specified in the *Library Exchange Format (.lef)* .

Silicon Ensemble completes placement and global routing and outputs the resultant design in the *Design Exchange Format (.def).* This output can then be directly used to determine the floorplanning information such as approximate wire routes and wire orderings, necessary for the formation of the composite buses within every gcell. The RT-level netlist is also passed through the RT-level profiler in order to get the data values on all the wires.

Once the profiling phase is complete, we *predict* the cross-coupling capacitances between wires in every gcell. The aggressor and victim resistances as well as their wire-to-ground capacitances are fixed for a given technology. In our case, $R_1 = 23.8\Omega$, $R_2 = 21.6\Omega$, $C_1 = 0.4$fF, and $C_2 = 0.2$fF. The dimensions of every gcell are also determined from the technology file. In all our experiments, each gcell is $18\mu m$ x $18\mu m$ and the target technology is 0.35μ. However, the proposed technique is independent of both the gcell dimensions as well as the target technology. Due to the absence of exact coordinates of the wire routes, the wires within a gcell are assumed to overlap for the entire length of the gcell. Experimental results show that this is a reasonable assumption in the post global routing phase and under this assumption, the peak amplitude estimates match the corresponding HSPICE simulations well.

5.5.1 RC models

In order to verify the results, we form RC models for every gcell through which a given wire passes. Since we model crosstalk as a pattern-dependent phenomenon, we simulate the model with the profiled data stream in HSPICE to compute the total number of crosstalk events that occur on the different wires. The total number of crosstalk events on a wire divided by the total length of the simulated data stream indicates the crosstalk probability

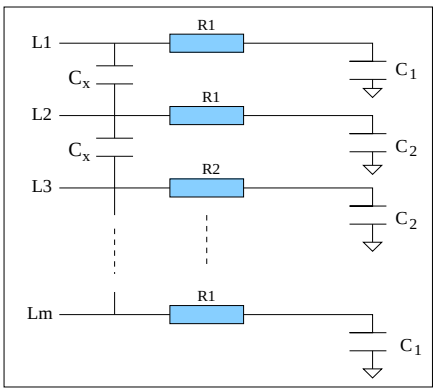


Figure 5.8. The RC model for every gcell

of the wire in the particular gcell. From the simulation waveform, we can also measure the peak noise amplitude attained by this wire in the gcell. The estimation errors are computed by comparing the estimated values against the simulation results. The RC models follow the conventional crosstalk model for bus-based interconnects, as shown in Figure 5.8.

In Figure 5.8, the gcell has m wires. Hence, the RC model has m lines, L_1 to L_m. The victim wire V is the third line L_3 in the RC model. Hence, the third wire has an associated lumped resistance $R2$ and a wire-to-ground capacitance $C2$. The remaining wires in the RC model have lumped resistances of $R1$ and wire-to-ground capacitances of $C1$. The cross-coupling capacitance is equal to C_X for all the wires.

Table 5.2 gives the benchmark details with respect to the number of wires present in the synthesized designs. The wires originate from nets belonging to functional units (FUs), registers (REG), multiplexers (MUX), and controller (CTRL). *Note that the number of wires is less because these nets are between macrocells. The macrocells themselves are placed and routed using a standard-cell place and route tool.*

Table 5.3 shows the accuracy of the statistical crosstalk probability estimation method within all the global routing cells in the different designs. The estimates are compared against the total number of crosstalk events obtained from the HSPICE simulation waveform. For a fixed length of the vector stimulus in the spice netlist, we can compute the corresponding crosstalk probability. Due to the large number of gcells, the maximum and

Table 5.2. Benchmark details

Benchmark	Wires				Total wires
	FUs	REG	MUX	CTRL	
DiffEq	16	27	16	15	74
FIR	42	80	48	18	188
IIR	49	80	64	20	213
FFT4pt	45	48	48	16	157
DCT2pt	90	96	80	26	292

Table 5.3. Crosstalk probability estimation errors compared to HSPICE

Benchmark	Max error	Min error	Avg error	Std deviation
DiffEq	31.7%	0%	7.1%	8.4%
FIR	33.3%	0%	1.4%	4.6%
IIR	34.3%	0%	2.3%	7.8%

minimum errors along with the the standard deviation of the errors are provided. The minimum error is zero for all the designs since the crosstalk probability is precisely estimated for a majority of the wires. The sources of error in this technique are as follows:

1. The empirical assumptions regarding the wire ordering. Although these are found to be true for a majority of cases, they are not true for all the gcells.

2. The estimation of the inter-wire coupling capacitance. We assume that all the wires are equally spaced inside a gcell. However, this may not be the case for all the gcells.

3. The intra-bus crosstalk estimator has some error because of the sampling procedure. This error is reflected in the inter-wire crosstalk estimation too.

Each victim wire can pass through global routing cells with one or more aggressors. Based on the total number of aggressors, various global routing cells are characterized. Estimates of the worst-case noise amplitude of a victim wire inside the different global routing cell models are compared against the corresponding HSPICE simulation. Figure 5.10 shows the simulation of a wire which passes through a gcell with a single aggressor. Thus, there are only two wires in this gcell - the victim wire and the aggressor wire. The

Table 5.4. Execution times of crosstalk probability estimation

Benchmark	Statistical Estimation	HSPICE
DiffEq	1.3min	2.6min
FIR	4.3min	2.3hr
IIR	95s	1.06hr

Table 5.5. Amplitude estimates against simulated values for different gcells

2 wires		3 wires		4 wires		5 wires	
Est.(mV)	Sim.(mV)	Est.(mV)	Sim.(mV)	Est.(mV)	Sim.(mV)	Est.(mV)	Sim.(mV)
20.5	20.0	30.1	25.0	39.4	42	48.2	45.0

estimated maximum amplitude for this global routing cell model is 20.5mV. Similarly, in Figure 5.11, the victim wire shares a gcell with three other wires. For this 4-wire gcell model, the estimated maximum amplitude is 39.4mV. It is to be noted that the amplitude for the second case is higher because the inter-wire separation for a larger number of wires routed through the same area is less and consequently, the cross-coupling capacitance is high. The rise and fall times of the inputs are fixed at 0.01ps for all experiments. All experiments were performed on a 128 MB Sun Ultra 2 dual-processor workstation.

For the current model where the statistical estimator is run once for each global routing cell through which a victim wire passes, the execution time comparisons between the estimator and HSPICE are provided in Table 5.10. The total execution time for a design is obtained by multiplying the crosstalk probability estimation time for a gcell by the product of the total number of wires in the design and the average number of gcells through which each wire passes. Thus although IIR has a larger number of wires than DiffEq, its execution time is lower because the average number of gcells through which each wire in IIR passes, is much lower than that of DiffEq.

Table 5.5 compares the analytically estimated worst-case crosstalk noise pulse amplitudes against the simulated values for global routing cells with two, three, four, and five neighbors. It may be noted that increasing number of neighbors leads to increased coupling capacitances which in turn, leads to increase in the peak noise amplitude.

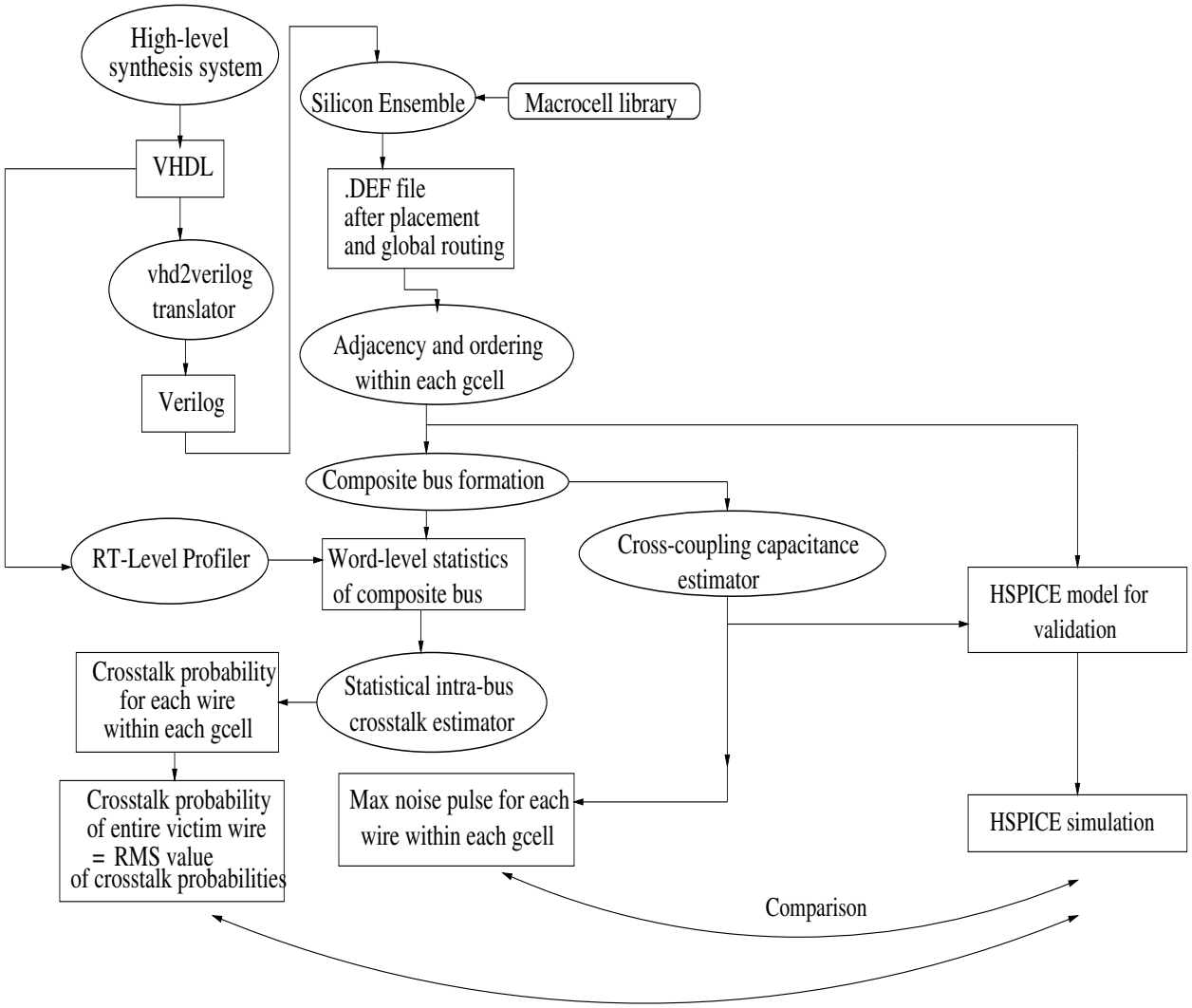


Figure 5.9. Experimental flow (uniform wire model)

Table 5.6. Crosstalk susceptibility distribution of victim nets (0.00 - 0.40)

Design	0.0-0.1		0.11-0.2		0.21-0.3		0.31-0.4	
	Stat	HSP	Stat	HSP	Stat	HSP	Stat	HSP
DiffEq	10%	0%	0%	0%	20%	20%	40%	40%
FIR	50%	40%	40%	40%	10%	20%	0%	0%
IIR	30%	30%	40%	30%	10%	10%	0%	0%
FFT4pt	10%	10%	40%	40%	10%	10%	30%	30%
DCT2pt	60%	60%	20%	20%	0%	0%	20%	20%

Table 5.7. Crosstalk susceptibility distribution of victim nets (0.41 - 0.70)

Design	0.41-0.5		0.51-0.6		0.61-0.7	
	Stat	HSP	Stat	HSP	Stat	HSP
DiffEq	10%	10%	20%	30%	0%	0%
FIR	0%	0%	0%	0%	0%	0%
IIR	0%	0%	10%	10%	10%	10%
FFT4pt	10%	10%	0%	0%	0%	0%
DCT2pt	0%	0%	0%	0%	0%	0%

Figure 5.9 shows the modification made to the existing technique to incorporate the uniform wire model. Table 5.8 shows the accuracy of the statistical crosstalk probability estimation method using the uniform wire model. The results are presented for the victim wires in the different designs. The estimates are compared against the total number of crosstalk events obtained from the HSPICE simulation waveform. Given a fixed length of the vector stimulus in the spice netlist, we compute the corresponding crosstalk probability. Due to the large number of nets for each design, we group them into 3 *error bins*, depending on the estimation errors with respect to the HSPICE simulations. For example, in the Finite Impulse Response filter, 50% of the victim nets have less than 10% estimation error. It may be observed that for all the designs, 80% or more of the victim nets have estimation errors less than 20%. Further, the maximum and minimum errors along with the average errors are provided in Table 5.9. The minimum error is zero for all the designs since the crosstalk probability is precisely estimated for a majority of the wires.

Table 5.8. Error bins of estimation errors wrt HSPICE (uniform wire models)

Benchmark	Estimation Error		
	$\leq$ 10%	10% - 20%	$>$ 20%
DiffEq	60%	30%	10%
FIR	50%	30%	20%
IIR	50%	30%	20%
FFT4pt	60%	20%	20%
DCT2pt	70%	10%	20%

Table 5.9. Estimation error statistics compared to HSPICE (uniform wire models)

Benchmark	Max err	Min err	Avg err
DiffEq	66.7%	0%	14.2%
FIR	33.3%	0%	10.0%
IIR	30.0%	0%	11.0%
FFT4pt	25.0%	0%	12.1%
DCT2pt	25.0%	0%	8.1%

The execution time comparisons between the statistical estimator and HSPICE are provided in Table 5.10. The analytical estimation process achieves a speedup of more than six times over that of HSPICE simulation.

5.6 Conclusions

We have presented a technique to estimate the crosstalk susceptibility of different nets in a design during physical-level synthesis. We used two metrics in order to measure the crosstalk susceptibility of each wire. The first was a crosstalk probability for the wire in

Table 5.10. Average execution times of estimation compared to simulation (each victim wire)

Benchmark	Statistical Estimation	HSPICE
DiffEq	46s	7.0min
FIR	41s	6.5min
IIR	33s	6.3min
FFT4pt	31s	6.8min
DCT2pt	48s	7.1min

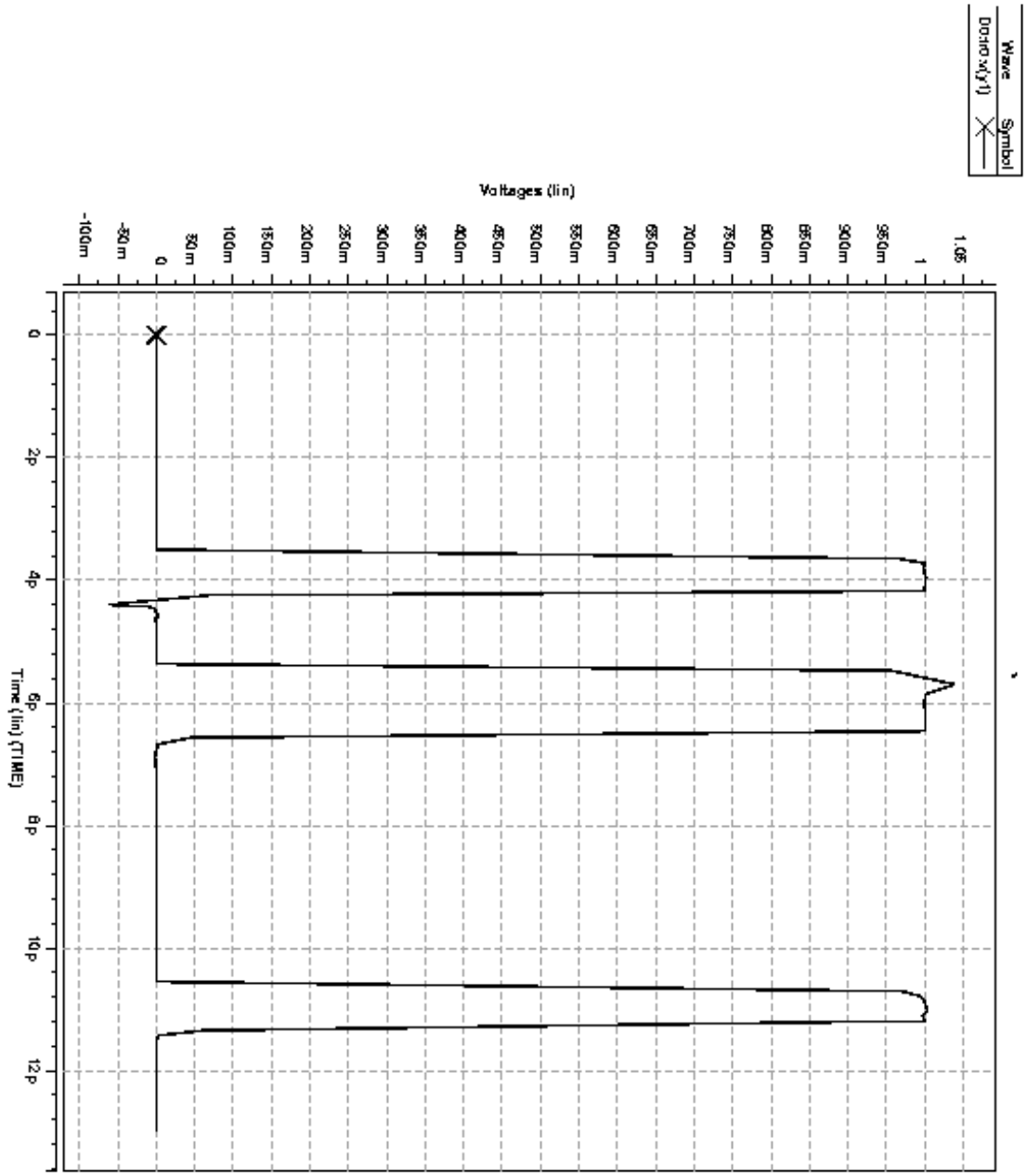


Figure 5.10. Amplitude simulation in HSPICE - 2-wire model

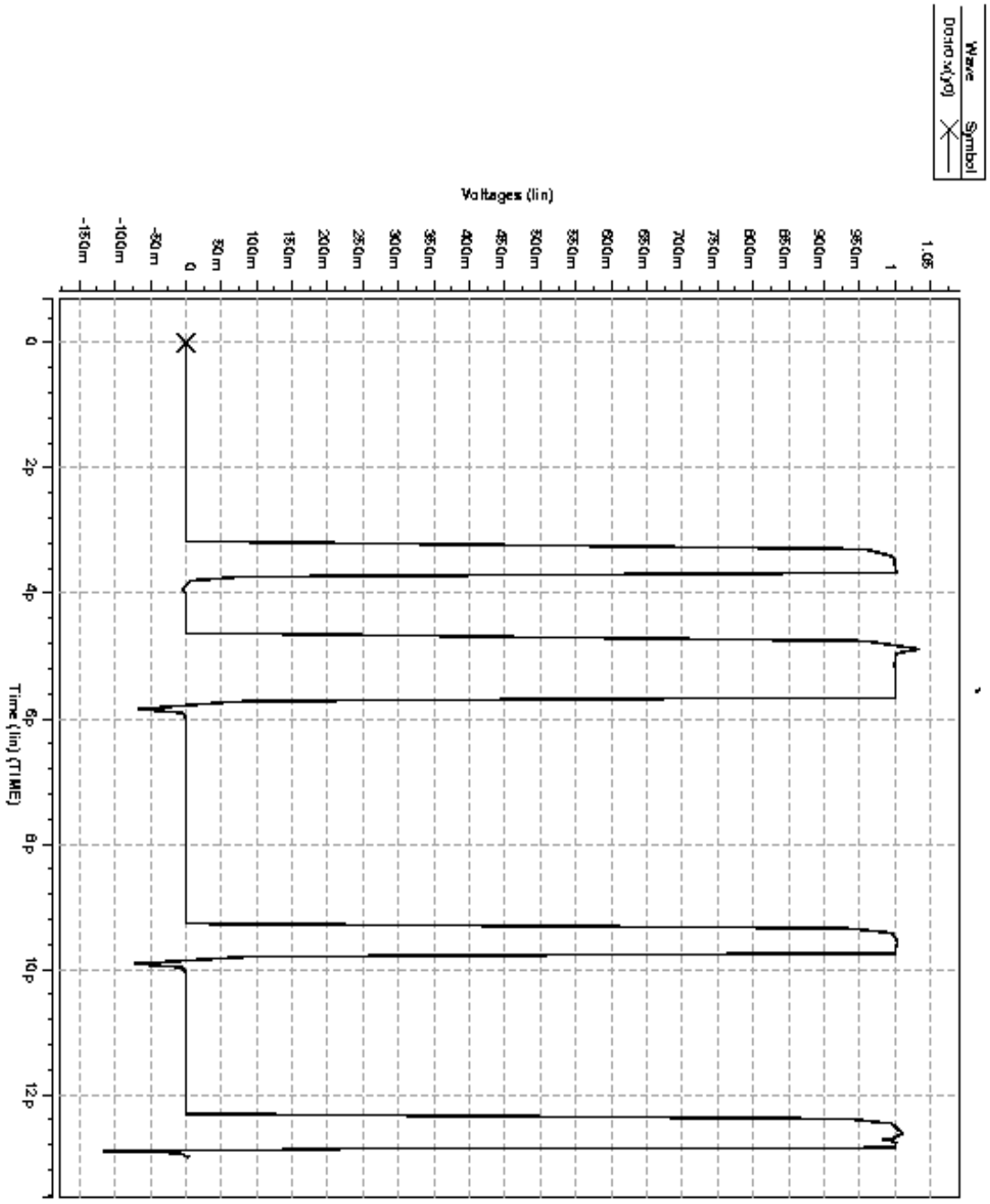


Figure 5.11. Amplitude simulation in HSPICE - 4-wire model

different regions along its route. This was evaluated using a fast and accurate statistical estimation procedure. The second was the worst case crosstalk noise pulse amplitude experienced by the wire along its route. For each metric, a series of values obtained for a given wire along its route is then combined into a single value for the entire wire. Comparison of both metrics against detailed HSPICE simulations show good accuracy with much shorter runtimes.

CHAPTER 6

BINDING FOR CROSSTALK MINIMIZATION DURING HIGH LEVEL SYNTHESIS

We utilize the crosstalk susceptibility information obtained for the victim nets of a design to direct the high-level synthesis process to produce an RT-level design that is more immune to crosstalk. The high-level synthesis process consists of three main steps namely, scheduling, allocation, and binding. Initially, we characterize different designs with respect to the crosstalk susceptibility of the inputs and outputs of their functional units. Based on the characterization, we formulate a cost function to evaluate the overall quality of the RT-level netlist with respect to crosstalk. We then modify the traditional clique partitioning algorithm by incorporating this cost function into it, so as to generate crosstalk-immune register bindings. Comparisons with regular register bindings which do not account for crosstalk, show reasonable crosstalk reduction.

6.1 The *au*tomatic *d*esign *i*nstantiation (*audi*) synthesis system

Before moving on to the details of the crosstalk optimization algorithm, it is necessary to discuss the foundation tool using which we perform high-level synthesis (HLS). This is the AUtomatic Design Instantiation (AUDI) system developed by our research group.

The AUDI system was developed to perform HLS on a design specified in the form of a data-flow graph. The data flow graph is written in a specific format known as the *audi instantiation format* (.aif). The .aif file for a simple example is shown below.

The file specifies the primary inputs and outputs of the design as well as the intermediate edges in the graph. Both the *edge-id* as well as the *edge-width* are specified for each edge. For example, the internal edge $t0$ has a width of 8. The inputs and outputs to the various

operations are specified in the order *left input id*, *right input id*, and *output id* from left to right. For example, operation 1 represents a multiplication. It has two inputs $x1$ and $x2$ and one output $t0$.

```
inputs x1 8 x2 8 x3 8 x4 8 x5 8 x6 8 x7 8 x8 8
outputs i 8
regs t0 8 t1 8 t2 8 t3 8 t4 8 t5 8
op1 MULT 8 x1 x2 t0
op2 MULT 8 x3 x4 t1
op3 MULT 8 x5 x6 t2
op4 MULT 8 t0 t1 t3
op5 MULT 8 t2 x7 t4
op6 SUB 8 t3 x8 t5
op7 SUB 8 t5 t4 i
end
```

AUDI then goes through an interactive sequence of steps and generates structural VHDL implementation of the datapath, controller, and the overall design. The steps involve choosing a suitable scheduling algorithm, specifying the number of resources to be used, and choosing the type of binding. If the number of resources is not specified, AUDI always uses the minimum number necessary to satisfy a given throughput. The information pertaining to the binding of operations to functional units and edges to registers is specified as a header in the structural VHDL files. The header information of the datapath file corresponding to the *.aif* file shown above is as follows.

```
--  File name:           simple4_dp.vhd
--  Type:                Datapath
--  Input aif file name: simple4.aif
--  CDFG statistics:
--    * Number of PI's:            8
```

```
--    * Number of PO's:             1
--    * Number of internal edges:   6
--    * Number of Operations:       7
--    * Conditionals:               -12345
--    * Loops:                      -12345
--    * Types of Operations:
--  Design Flow/Algorithm Information:
--    * Scheduling:        ASAP
--    * Allocation:        Automatic
--    * Binding:           Automatic
--    Interconnect style:  Mulitplexor-based
--  Design Information:
--    Datapath:
--     * Registers:          8
--     * Functional units:   4
--     * Number of Muxes:    5
--     * Number of Buses:    0
--     * Operator Binding Information:
-- Resource Id=0 type = MULT :
--  Index = 0  type= MULT width = 8
-- Mapped Ops = {  op1  op4  }
--  Index = 1  type= MULT width = 8
-- Mapped Ops = {  op2  op5  }
--  Index = 2  type= MULT width = 8
-- Mapped Ops = {  op3  }
-- Resource Id=1 type = SUB :
--  Index = 0  type= SUB width = 8
-- Mapped Ops = {  op6  op7  }
```

```
--     * Register Optimization Information:
-- Register #0 (width = 8, ctrl = -12345) =
--   {  i  t5  t3  t0  x1  }
-- Register #1 (width = 8, ctrl = -12345) =
--   {  t4  t1  x2  }
-- Register #2 (width = 8, ctrl = -12345) =
--   {  t2  x3  }
-- Register #3 (width = 8, ctrl = -12345) =
--   {  x8  }
-- Register #4 (width = 8, ctrl = -12345) =
--   {  x7  }
-- Register #5 (width = 8, ctrl = -12345) =
--   {  x6  }
-- Register #6 (width = 8, ctrl = -12345) =
--   {  x5  }
-- Register #7 (width = 8, ctrl = -12345) =
--   {  x4  }
--    Controller:
--     * Type:                    Moore
--     * Number of states:         7
--     * Number of control bits:   14
-------------------------------------------------------------------
```

For example, operations 1 and 4 which are both multiplications, are bound to the same multiplier instance. Similarly, internal edges $t4$, $t1$, and primary input $x2$ are bound to the same register namely, Register #1.

During high-level synthesis, AUDI is capable of performing design optimizations such as dynamic power optimization [45] and leakage power optimization [88][89][90][91]. We introduce crosstalk optimization as an additional feature into the system.

6.2 Crosstalk characterization of designs

Using AUDI, we make a preliminary characterization of designs with respect to crosstalk in their datapaths. The datapath crosstalk is sub-divided into the following categories:

- Crosstalk of the register outputs
- Crosstalk of the functional unit outputs
- Crosstalk of the multiplexor outputs

The metric used for measuring crosstalk is the *crosstalk probability* between lines of the various datapath nets. For example, while computing the amount of crosstalk of the register outputs, we compute the crosstalk susceptibility of each line at the output of every register in the design. If the total number of registers being used is n_{reg} and each register is m-bits wide, we compute a set of crosstalk coefficients corresponding to the different bits. For each bit, the crosstalk coefficient is the average crosstalk activity of that particular bit across all the registers. For example, we compute the Least Significant Bit (LSB) coefficient as the average crosstalk activity on the LSB of all the registers. Similarly, we compute the crosstalk coefficient of each of the other bits. This rationale for comparing the same bit across all the registers is derived from the Dual-Bit model [9] where the entire word is grouped into different regions based on breakpoints. Bits in the same region experience similar activity. Thus, all the LSBs of the different register outputs will approximately experience the same amount of crosstalk activity and their average is a good estimate of the activity. Thus, for m-bit registers, there will be m coefficients. In general, the crosstalk coefficient corresponding to the ith bit i.e. $C_{i,reg}$, is defined as follows:

$$C_{i,reg} = \frac{\sum_{k=1}^{n_{reg}} c_{i,k}}{m} \tag{6.1}$$

Similarly, we compute the amount of crosstalk in the functional units and multiplexors.

The characterization is made for three different scheduling algorithms, which are as follows:

Table 6.1. Details of DiffEq datapath

Scheduling	Registers	Functional Units	Multiplexors
ASAP	8	4	5
ALAP	8	3	7
FDS	8	3	7

Table 6.2. DiffEq datapath characterization - asap

Bus line	Registers	Functional Units	Multiplexors
1	5%	6%	9%
2	6.9%	11.4%	14.2%
3	6.7%	12.3%	14.3%
4	6.1%	13.2%	13.7%
5	4.4%	12.1%	10.8%
6	2.8%	10.1%	7.8%
7	1.5%	7.1%	4.6%
8	1.0%	4.7%	3.2%

- The unconstrained As-Soon-As-Possible (ASAP) algorithm
- The latency-constrained As-Late-As-Possible (ALAP) algorithm
- the latency-constrained Force-Directed Scheduling algorithm (FDS) which minimizes resource usage [92]

Tables 6.1-6.9 report the datapath details and the characterization results for the different designs. The characterization entries indicate the average crosstalk activity for each edge of the 8-bit resources. For example, in Table 6.2, the LSB of the registers have 5% crosstalk activity on the average while the MSB of the registers have 1% crosstalk activity on the average.

6.3 Binding during high-level synthesis

Following scheduling and allocation of resources, the last step in our high-level synthesis flow is *binding*. This consists of mapping different operations and edges in the data flow graph to the available resource instances. The binding step is subdivided into the following:

Table 6.3. DiffEq datapath characterization - alap/fds

Bus line	Registers	Functional Units	Multiplexors
1	5.1%	10.5%	13.5%
2	7.1%	18.0%	20.8%
3	7.0%	18.9%	21.8%
4	6.2%	20.3%	21.0%
5	4.5%	19.1%	18.3%
6	3.2%	17.7%	14.5%
7	1.9%	10.9%	8.8%
8	1.3%	9.8%	6.5%

Table 6.4. Details of FIR datapath

Scheduling	Registers	Functional Units	Multiplexors
ASAP	10	6	6
ALAP	10	3	6
FDS	10	3	6

Table 6.5. FIR datapath characterization - asap

Bus line	Registers	Functional Units	Multiplexors
1	4.9%	8.5%	11.6%
2	7.7%	13.8%	18.6%
3	7.1%	14.6%	19.0%
4	6.8%	14.9%	18.5%
5	4.7%	13.7%	16.6%
6	3.1%	10.9%	12.3%
7	1.3%	6.4%	6.1%
8	0.0%	3.6%	3.9%

Table 6.6. FIR datapath characterization - alap/fds

Bus line	Registers	Functional Units	Multiplexors
1	4.9%	15.0%	14.3%
2	7.5%	23.3%	22.6%
3	6.8%	24.7%	22.1%
4	6.5%	25.6%	21.8%
5	4.0%	23.4%	17.9%
6	2.8%	20.9%	13.5%
7	1.3%	12.5%	7.1%
8	0.0%	6.6%	3.9%

Table 6.7. Details of IIR datapath

Scheduling	Registers	Functional Units	Multiplexors
ASAP	10	7	8
ALAP	10	5	10
FDS	10	5	10

Table 6.8. IIR datapath characterization - asap

Bus line	Registers	Functional Units	Multiplexors
1	6.6%	7.0%	13.8%
2	11.1%	12.7%	23.7%
3	10.5%	12.7%	23.8%
4	11.3%	13.7%	26.1%
5	9.4%	13.9%	25.7%
6	9.5%	12.2%	23.5%
7	4.5%	9.5%	16.4%
8	2.3%	7.5%	9.5%

Table 6.9. IIR datapath characterization - alap/fds

Bus line	Registers	Functional Units	Multiplexors
1	5.6%	11.4%	14.8%
2	9.3%	18.4%	24.3%
3	8.2%	18.0%	23.0%
4	8.2%	18.0%	23.5%
5	6.4%	17.1%	20.1%
6	5.9%	15.9%	19.3%
7	3.4%	9.7%	10.8%
8	1.9%	8.2%	8.5%

- Binding of edges to registers.
- Binding of operations to functional unit instances.

The binding of operations to a minimum number of functional units is simple. Based on the scheduling algorithm selected, the operations in the data-flow graph have data dependencies between them. Figure 6.1 shows an example of two different schedules for the same data flow graph [93]. Thus, in the ASAP schedule shown, operations 3 and 7 are scheduled in the same timestep (tstep) and may not share the same multiplier instance. However, in the ALAP schedule, they may do so since they are scheduled in different timesteps. Similarly, operations 7 and 8 which may share a multiplier instance in the ASAP schedule, may not do so in the ALAP schedule. The ASAP schedule needs a minimum of four multipliers and two ALUs while the ALAP schedule needs a minimum of two multipliers and three ALUs. In order to minimize the number of functional units used, the strategy is to keep binding different operations to the same functional unit instance as long as data dependencies are not violated. This is a greedy approach and works well for small to medium-sized benchmarks.

The binding of edges to registers is however, more involved. It first determines the set of *compatible* pairs of edges. Two edges are said to be compatible if they have non-overlapping lifetimes i.e., if they can potentially share a register. Based on the edge compatibilities, a *register compatibility graph* is formed. The vertices represent the edges in the data flow graph and two vertices are connected if the edges corresponding to those are compatible [94][95][96].

The next step involves forming *cliques* in the register compatibility graph. A clique is defined as a subset of vertices in the graph, all of which are mutually connected by edges. In the register compatibility graph, a clique will be formed by edges which are mutually compatible.

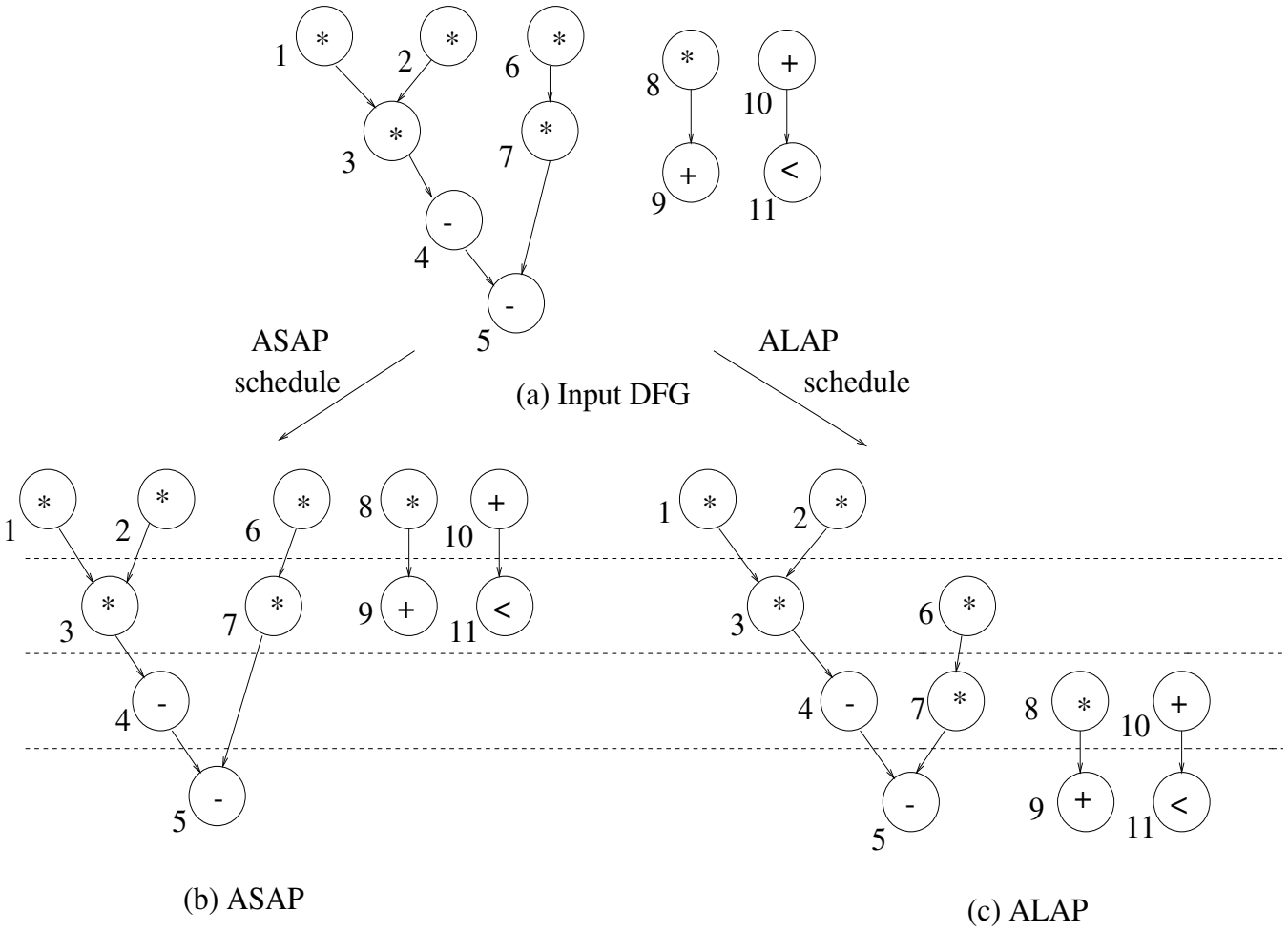


Figure 6.1. Asap/alap schedules for a data flow graph

6.3.1 Clique partitioning

Finding the maximal clique in a given graph is a NP-hard problem. Among the heuristics proposed to solve it, Tseng and Sieworek's heuristic [97] has been the most widely applied to high-level synthesis. The algorithm is illustrated using Figure 6.2.

Starting with the initial compatibility graph having six vertices, the first step of the Tseng-Sieworek algorithm chooses a pair of vertices having the highest number of common neighbors as seed. This is the pair of vertices *v1, v3* which have two common neighbors namely, *v8* and *v7*. Thus, *v1, v3* is chosen as the seed. Vertices *v1* and *v3* are now merged into a common vertex. If a vertex was previously connected to both *v1* and *v3*, a single edge now connects it to the merged vertex *v1, v3*. If a vertex was previously connected to either *v1* or *v3* but not both, it is no longer connected to the merged vertex. Thus, in the new graph with five vertices, edge *v1, v6* is removed because there was no connection from *v3* to *v6* in the initial graph. The next step of the algorithm selects vertex *v7* since it is connected to vertex *v1, v3*. The subset of vertices *v1, v3, v7* is then identified as a clique. The algorithm then searches for a new vertex pair as the seed for the next clique. Whenever there is more than one option, the algorithm selects one randomly. Suppose it picks the pair *v6, v8* as the next seed. In the next step, *v2* is added to form the second clique *v6, v8, v2* [94].

While applying this algorithm to determine the edges which may share a register in the datapath (without crosstalk considerations), the current clique picks a compatible vertex which shares the maximum number of neighbors with it. If there is more than one possibility, the vertex whose selection would *exclude the minimum number* of neighbors of the clique, is selected. This is to ensure maximality of the clique.

Similarly, while binding for crosstalk minimization, the current clique always picks a compatible vertex sharing the maximum number of vertices with it. However, to break a tie between two or more vertices, it interleaves the data streams of the vertices which are part of the current clique and each of the vertices which are candidates for selection in the current iteration. The crosstalk activity for each selection is then computed. The

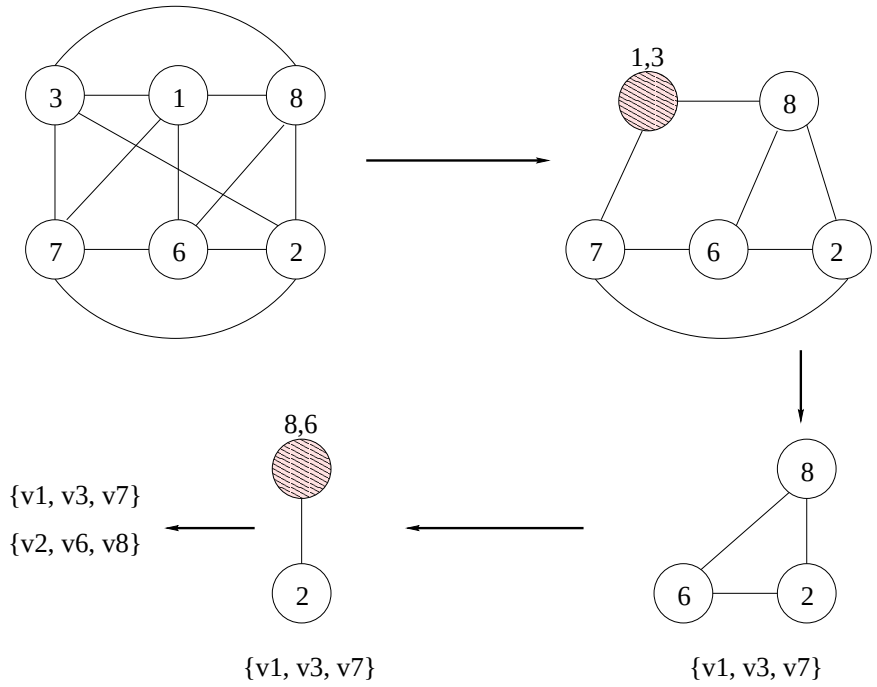


Figure 6.2. Clique partitioning example

vertex which results in the lowest crosstalk activity for the new clique, is finally selected. Figure 6.3 illustrates the procedure.

In the figure, *v1, v3* are part of the current clique. The algorithm currently has a choice of three vertices namely, *v2*, *v5*, and *v7* for expanding the clique. The usual procedure is to choose one vertex randomly. However, for minimizing crosstalk, we interleave the data stream of the current clique with each candidate vertex and compute the crosstalk activity, as shown in the figure. The vertex corresponding to the lowest crosstalk activity is chosen. Suppose Y is the lowest value. Then, we choose *v5* as the candidate vertex and update the clique.

6.4 Experimental flow and results

The experimental flow is shown in Figure 6.4. Each design is input to the AUDI system in the *audi intermediate format* (.aif) format. Of several possible scheduling algorithms available in AUDI, we restrict our choices to the ASAP, ALAP, and FDS scheduling algo-

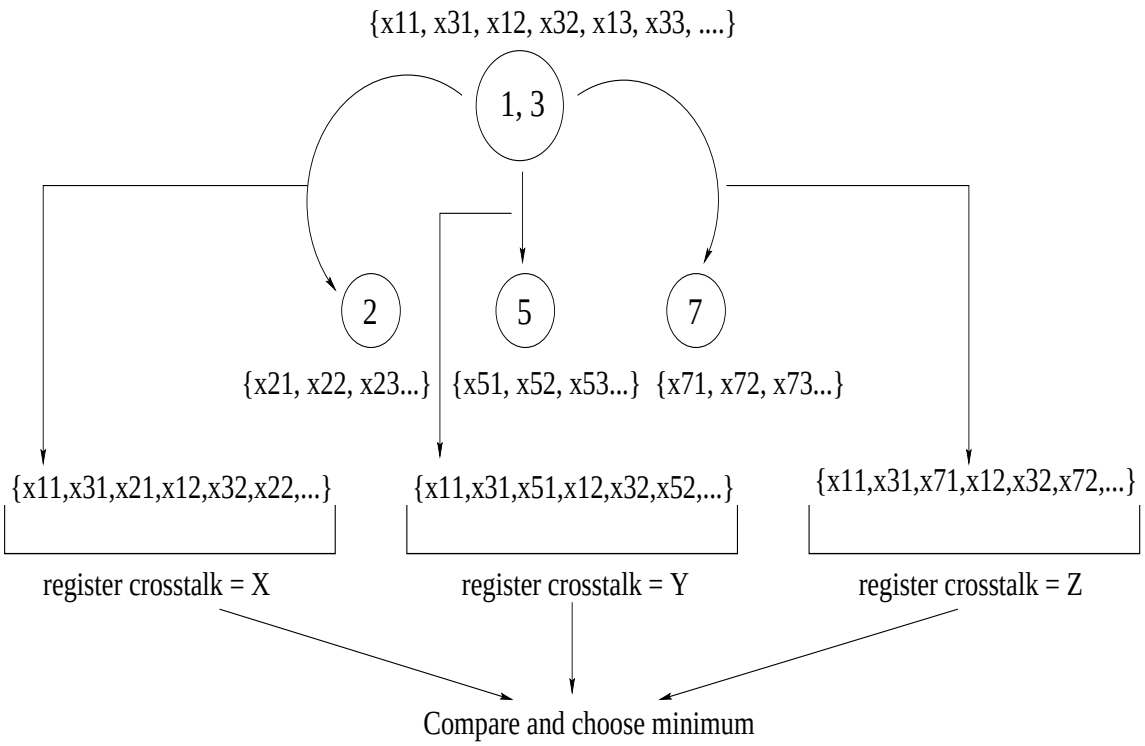


Figure 6.3. Clique partitioning for crosstalk minimization

rithms since our main objective is to explore the binding space for crosstalk minimization. Scheduling is followed by resource allocation and then by binding. Binding may be performed in the *regular* manner without taking crosstalk activity into account. On the other hand, we can program the system to perform *crosstalk-aware* binding which minimizes crosstalk activity in the registers. For the same scheduling algorithm, the generated RTL datapaths are different for different bindings. Each datapath is profiled with data streams from different data environments using the RT-Level profiler that we previously developed. The RTL designs are then simulated using Cadence nclaunch and the crosstalk activity at the outputs of the different registers is computed and compared.

Tables 6.10 - 6.12 compare the regular and the crosstalk-aware bindings of different designs in terms of the register crosstalk activity. The scheduling algorithm chosen for both bindings is the same so that we are able to capture only the effect of the crosstalk-aware binding on the register crosstalk activity. The scheduling algorithm chosen for each design

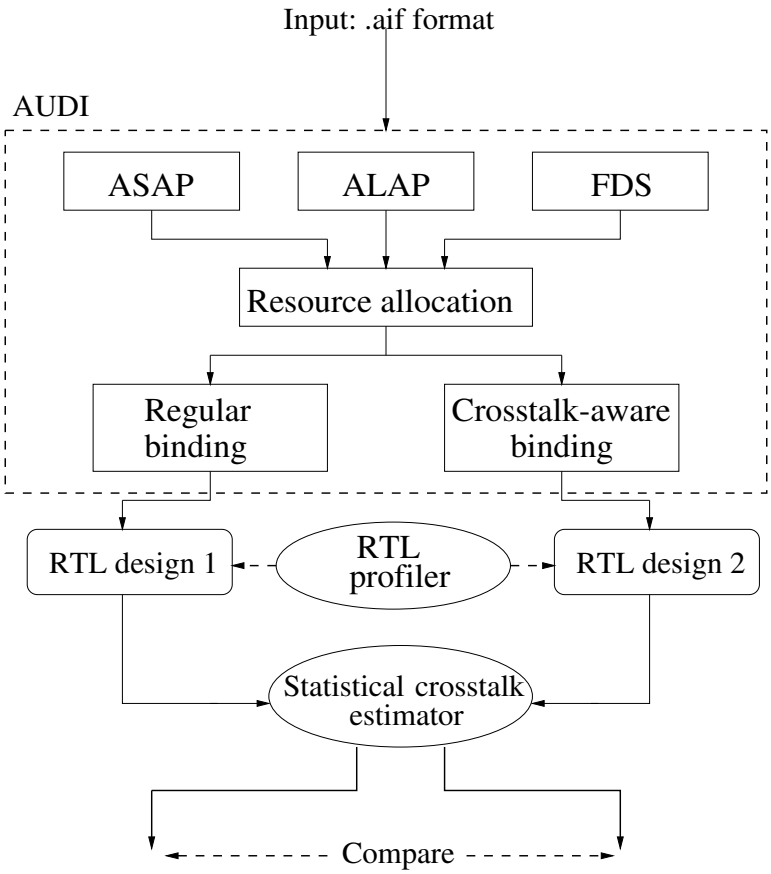


Figure 6.4. Experimental flow for rtl crosstalk optimization

Table 6.10. Crosstalk reduction due to crosstalk-aware binding (asap scheduling) - DiffEq

Register ID	Crosstalk reduction (%)
1	2.0%
2	0.0%
3	0.0%
4	0.0%
6	0.0%
7	1.0%
8	12.0%

Table 6.11. Crosstalk reduction due to crosstalk-aware binding (asap scheduling) - FIR filter

Register ID	Crosstalk reduction (%)
1	4.1%
2	12.2%
3	6.3%
4	16.4%
5	0.0%
6	-1.7%
7	7.2%
8	7.9%
9	0.0%
10	0.0%

is the one which gives the lowest crosstalk activity with the regular binding. We obtain this information from the initial characterization of the design. Thus, we compare the proposed crosstalk-aware binding technique against the regular binding for the minimum crosstalk-activity schedule.

From Tables 6.2 and 6.3, it can be seen that the crosstalk activity in the datapath for the DiffEq example is lesser in case of the ASAP schedule. Hence, in Table 6.10, we test the effectiveness of the crosstalk-aware binding against the regular binding for the ASAP schedule. Since the DiffEq example has eight primary inputs, AUDI uses only eight registers, which is the minimum number required for a feasible design. The percentage reduction in crosstalk due to the difference in the regular and crosstalk-aware bindings is reported register by register for each design. Similarly, in Tables 6.11 and 6.12, we choose

Table 6.12. Crosstalk reduction due to crosstalk-aware binding (alap scheduling) - IIR filter

Register ID	Crosstalk reduction (%)
1	2.0%
2	4.5%
3	5.1%
4	4.2%
5	0.0%
6	1.5%
7	2.0%
8	9.0%
9	2.0%
10	3.4%

Table 6.13. Comparion of runtimes

Design	Schedule	Regular binding	Crosstalk-aware binding
DiffEq	ASAP	3s	59s
FIR	ASAP	2s	59s
IIR	ALAP	2s	57s

the schedule which is the lowest in terms of crosstalk activity for each design and test the proposed crosstalk-aware binding against the regular binding for the chosen schedule.

Table 6.13 compares the runtimes for the entire high-level synthesis process for the regular and crosstalk-aware binding choices. The crosstalk-aware binding takes more time because of the interleaving of data streams and computing the crosstalk activity while choosing a candidate vertex for expanding the clique. Tables 6.14 compare the resource usage for a common schedule and both types of binding, for each design.

Table 6.14. Comparion of resource usage

		Regular binding			Crosstalk-aware binding		
Design	Schedule	Reg	FU	Mux	Reg	FU	Mux
DiffEq	ASAP	8	4	5	8	4	6
FIR	ASAP	10	6	6	10	6	6
IIR	ALAP	10	5	10	10	5	12

6.5 Conclusions

We have presented a crosstalk-aware register binding technique during high-level synthesis that minimized crosstalk activity at the register outputs in the RT-level design. We initially characterized the designs with respect to three different scheduling algorithms. Using the schedule which favors lower crosstalk activity, we compared the proposed crosstalk-aware binding with the regular clique partitioning-based binding. Reductions in register crosstalk activity of over 16% were obtained over various designs. Integration of the proposed binding algorithm with a constructive crosstalk-aware scheduling has the potential to minimize the crosstalk activity at the outputs of all the datapath units and is part of the future work.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The current trends in VLSI result in new and complex problems with respect to circuit design. Miniaturization of devices for portability and the continuing effects of Moore's law create unique interactions between the device components which affect the overall performance and reliability. Crosstalk is one such phenomenon caused by the proximity of the interconnect wires. Unwanted interactions between these wires affect both the reliability and performance of the circuit adversely. Thus, it becomes imperative for the designer to accurately estimate and optimize crosstalk in order to generate error-free designs.

Due to the advantage of fast design space exploration at the higher levels of design abstraction and the proven speed and accuracy of statistical models in particular, this dissertation proposed fast and accurate statistical estimators of crosstalk to estimate the crosstalk susceptibility of the lines of bus-based interconnects. The estimator took only the word-level statistics of the data on the bus as its input and computed bit-level crosstalk probabilities for each line. Comparisons with detailed HSPICE simulations indicated speedups of 10x or more while maintaining reasonable accuracy. The statistical estimator was further modified to linearize its complexity with respect to the bus-width and enhance its scalability. The modified estimator resulted in speedups of over two orders of magnitude compared to HSPICE. The drawback in this technique is its tradeoff between speed and accuracy. The technique can be made more accurate but at the cost of increased runtimes. The number of samples specified by the user during the trapezoidal integration is, in fact, one of the sources of error in the technique. Moreover, this technique considered ideal bus geometries.

With the intra-bus crosstalk estimator in place, we then used it to determine the crosstalk susceptibility of different buses at the layout level of design abstraction. This

was a more challenging problem, considering the fact that we needed accurate physical-level information on the interconnect lines which was not available to us at the high-level. To overcome this, we used the Cadence Silicon Ensemble Place and Route tool to perform placement and global routing on the designs. Besides, we used the Cadence NCLaunch tool to perform RT-Level profiling on the designs and obtain the word-level values on the wires in the layout. The route of a victim wire was then split up into a collection of global routing cells (gcells) and the previously developed intra-bus estimator was run on each gcell. We also used analytical equations to compute the worst-case crosstalk noise pulse on the victim line within each gcell. Subsequently, the individual gcell values were combined to form a single value for the probability of crosstalk on a victim line as well as the worst-case noise amplitude on it, during its route. The sources of error in this technique are the differences in our predicted estimates of the physical-level parameters such as coupling capacitance and wire route, and the actual values of these parameters which are a function of the place and route tool. With more information from the circuit layout-level, the technique can be made more accurate.

Finally, we used the crosstalk probability metric to minimize crosstalk activity at the register outputs of a design during high-level synthesis (HLS). Initially, we characterized the crosstalk activity in designs with respect to different scheduling algorithms. For a given schedule, we then searched the binding space during the HLS process to find a suitable binding which minimized crosstalk at the output of the registers. In order to do this, we modified the traditional clique-partitioning algorithm to select nodes with minimum crosstalk activity during each iteration of the algorithm. The main drawback of this technique is its *locality* i.e. it is a greedy heuristic which explores the binding space alone. Exploring the entire high-level synthesis space can result in better solutions for decreasing the crosstalk activity in the entire datapath.

The research done in this work for tackling the crosstalk estimation and optimization problem leaves room for future work. Some of them are listed below.

- During the intra-bus crosstalk estimation, we assume that all the wires in a bus are equally spaced and run parallel for the entire length of the bus. However, such a perfect geometry may not be realized all the time. Hence, the statistical estimation algorithm may need to be modified to account for irregularities in the bus structure.

- While estimating the crosstalk effects on victim nets at the layout-level, an intrinsic assumption made by us was the availability of the values on all the aggressor wires inside a given routing channel. This assumption may need to be modified in view of the fact that each circuit has some timing associated with it. Thus, if a given victim has two adjacent aggressors on either side of it, it may be possible that only one of them has a value while the other one may be in the high-impedance state. It may also be possible that all the aggressors have values on them at some instant. Thus, specific *timing windows* may need to be associated with each routing channel.

- Since the steps of scheduling, allocation, and binding during high-level synthesis are inter-dependent, solving only one of them for crosstalk may result in a sub-optimal solution at times. Thus, if the crosstalk-aware binding algorithm proposed by us is integrated with crosstalk-aware scheduling, the crosstalk activity at the outputs of the functional units and multiplexors can be significantly reduced as well.

REFERENCES

[1] T. K. Tien, S. C. Chang, and T. K. Tsai. "Crosstalk Alleviation for Dynamic PLAs". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 21(12):1416–1424, December 2002.

[2] K. W. Kim, S. O. Jung, U. Narayanan, C. L. Liu, and S. M. Kang. "Noise-aware interconnet power optimization in domino-logic synthesis". *IEEE Transactions of Computer-Aided Design of Integrated Circuits and Systems*, 11(1):79–89, February 2003.

[3] . The international technology roadmap for semiconductors. In *Semiconductor Industry Association*, 2003.

[4] L. Green. "Simulation and Modeling - Understanding the importance of signal integrity". In *Circuits and Devices*, pages 7–10, 1999.

[5] R. Dean Adams. *"High performance memory testing design principles, fault modeling, and self-test"*. Kluwer Academic Publishers, 2002.

[6] K. P. Parker. *"Integrated Design and Test: Using CAE tools for ATE programming"*. IEEE Computer Society Press, 1987.

[7] N. Sherwani.,. *"Algorithms for VLSI Physical Design Automation: Third Edition "*. Kluwer Academic Publishers, 1999.

[8] M. Abramovici, M. A. Breuer, and A. D. Friedman.,. *"Digital Systems Testing and Testable Design"*. John Wiley and sons, 1995.

[9] P. Landman and J. M. Rabaey. "Architectural Power Analysis : The Dual Bit Type Method". *IEEE Transactions on Very Large Scale Integrated Systems*, 3(2):173–187, June 1995.

[10] P. Landman and J. Rabaey. "Power Estimation for High Level Synthesis". In *Proceedings of European Design Automation Conference*, pages 361–366, Paris, February 1993.

[11] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj. "Analytical Estimation of Signal Transition Activity from Word-Level Statistics". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 16(7):718–733, July 1997.

[12] J. H. Satyanarayana, and K. K. Parhi. "Theoretical Analysis of Word-Level Switching Activity in the Presence of Glitching and Correlation". *IEEE Transactions on VLSI*, April 2000, 8(2).

[13] A. Devgan. "Efficient coupled-noise estimation for on-chip interconnects". In *Proceedings of International Conference on Computer-Aided Design*, pages 147–153, 1997.

[14] M. Kuhlmann and S. Sapatnekar. "Exact and Efficient Crosstalk Estimation". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 20(7):858–866, July 2001.

[15] C. Y. Chu and M. A. Horowitz. "Charge-sharing models for switch-level simulations". *IEEE Transactions of Computer-Aided Design of Integrated Circuits and Systems*, 6(6):1053–1061, November 1987.

[16] A. Vittal and M. M. Sadowska. "Crosstalk Reduction for VLSI". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 16(3):290–298, March 1997.

[17] A. Vittal, L. H. Chen, M. M. Sadowska, K. P. Wang, and S. Yang. "Modelling crosstalk in resistive interconnections". In *Proceedings of 12th International Conference on VLSI Design*, pages 470–475, 1999.

[18] A. Vittal, L. H. Chen, M. M. Sadowska, K. P. Wang, and S. Yang. "Crosstalk in VLSI Interconnections". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 18(12):1817–1824, December 1999.

[19] N. Weste and K. Eshraghian. *"Principles of CMOS VLSI Design: A Systems Perspective"*. Addison-Wesley, 1985.

[20] S. M. Kang and Y. Leblebici. *"CMOS Digital Integrated Circuits Analysis and Design"*. WCB/McGraw-Hill, 1999.

[21] K. Shepard and V. Narayanan. "Noise in submicron digital design". In *Proceedings of International Conference on Computer-Aided Design*, pages 524–531, November 1996.

[22] P. C. Elmendorf G. Zheng K. Shepard, V. Narayanan. "Global Harmony: Coupled noise for full-chip RC interconnect networks". In *Proceedings of International Conference on Computer-Aided Design*, pages 139–146, November 1997.

[23] C. K. Tsai and M. M. Sadowska. "Modeling crosstalk-induced delay". In *Proceedings of Fourth International Symposium on Quality Electronic Design (ISQUED)*, page , 2003.

[24] K. M. Buyuksahin and F. N. Najm. "High-Level Power Estimation with Interconnect Effects". In *International Symposium on Low Power Electronics and Design*, pages 271–274, 2000.

[25] Z.J. Lemnois, and K.J. Gabriel. "Low-Power Electronics". *Design & Test of Computers*, pages 8–13, Winter 1994.

[26] J. Cong, C-K. Koh, and K-S.Leung. "Simultaneous driver and wire sizing for performance and power optimization". *IEEE Transactions on Very Large Scale Integrated Systems*, 2(4):408–425, December 1994.

[27] A. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Brodersen. "HYPER-LP: A System for Power Minimization Using Architectural Transformations". In *Proceedings of International Conference on Computer Aided Design*, pages 300–303, 1992.

[28] A. Chandrakasan et. al.,. "Optimizing Power Using Transformations". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, January 1995.

[29] C M Huizer. "Power Dissipation analysis of CMOS VLSI circuits by means of switch level simulation". In *IEEE European Solid State Ciruits Conference*, pages 61–64, Grenoble, France, 1990.

[30] J-M. Chang and M. Pedram. "Low Power Register Allocation and Binding". In *32nd Design Automation Conference*, pages 29–35, 1995.

[31] D. Liu and C. Svensson. "Power Consumption Estimation in CMOS VLSI Chips". *IEEE Journal on Solid State Circuits*, 29(6):663–669, June 1994.

[32] M. Pedram. "Power Minimization in IC Design: Principles and Applications". *ACM Transactions on Design Automation of Electronic Systems*, Vol. 1, No. 1, pp:3–56, January 1996.

[33] J. Y. Choi, C. H. Lin, H. S. Kim. "A low power register scheduling and allocation algorithm for multiple voltage". In *Proceedings of IEEE Region 10 Electrical and Electronic Technology*, pages 627–630, 2001.

[34] S. Hua, G. Qu. "Approaching the maximum energy saving on embedded systems with multiple voltages". In *International Conference on Computer-Aided Design*, pages 26–29, 2003.

[35] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri. "Profile-Driven Behavioral Synthesis for Low Power VLSI Systems". *IEEE Design & Test of Computers*, pages 70–84, Fall 1995.

[36] S. Katkoori, N. Kumar, L. Rader, and R. Vemuri. "A Profile Driven Approach for Low Power Synthesis". In *Proceedings of VLSI, Japan*, pages 759–765, 1995.

[37] S. Katkoori, N. Kumar, and R. Vemuri. "High Level Profiling Based Low Power Synthesis Technique". In *Proceedings of International Conference on Computer Design*, pages 759–765, October 1995.

[38] S. Katkoori and R. Vemuri. "Architectural Power Estimation Based on Behavioral Profiling". Special Issue on Low Power Design, Journal on VLSI Design, 1998.

[39] C. B. Shung, R. Jain, K. Rimey, E. Wang, M. B. Srivastava, B. C. Richards, E. Lettang, S. K. Azim, L. Thon, P. N. Hilfinger, J. M. Rabaey, R. W. Brodersen. "An integrated CAD system for algorithm-specific IC design". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(4):447–463, April 1991.

[40] N. Ranganathan S. P. Mohanty and V. Krishna. "Datapath scheduling using dynamic frequency clocking". In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 58–63, 2002.

[41] S. Raje and M. Sarrafzadeh. "Variable Voltage Scheduling". In *Proceedings of ISLPD*, pages 9–14, April 1995.

[42] M. Sarrafzadeh and S. Raje. "Scheduling with multiple voltages under resource constraints". In *Proceedings of the 1999 International Symposium on Circuits and Systems*, pages 350–353, June 1999.

[43] N. Ranganathan S. P. Mohanty and S. K. Chappidi. "ILP models for energy and transient power minimization during behavioral synthesis". In *Proceedings of the 17th International Conference on VLSI Design*, pages 745–748, 2004.

[44] N. Ranganathan S. P. Mohanty and S. K. Chappidi. "Transient power minimization through datapath scheduling in multiple supply voltage environment". In *Proceedings of 10th IEEE International Conference on Electronics, Circuits, and Systems*, pages 300–303, 2003.

[45] S. Gupta and S. Katkoori. "Force-directed scheduling for dynamic power optimization". In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 68–73, 2002.

[46] F. N. Najm. "Transition Density: A Stochastic Measure of Activity in Digital Circuits". In *DAC*, pages 644–649, 1991.

[47] A.M. Hill, C-C. Teng, and S-M. Kang. "Estimation of maximum transition counts at internal nodes in CMOS VLSI circuits". In *1996 IEEE International Symposium on Circuits and Systems*, volume 4, pages 13–16, 1996.

[48] A. K. Murugavel and N. Ranganathan. "A game theoretic approach for power optimization during behavioral synthesis". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(6):1031–1043, December 2003.

[49] J. Sairamesh D. F. Ferguson, C. Nikolaou and Y. Yemini. "Economic models for allocating resources in computer systems". In *A Paradigm for Distributed Resource Allocation, S. Clearwater Ed.*, 1996.

[50] J. F. Nash. "Noncooperative games". *Annals of Mathematics*, 54(2):286–295, September 1951.

[51] P.G. Paulin and J.P. Knight. "Force-Directed Scheduling in Automatic Data Path Synthesis,". In *24th Design Automation Conference*, pages 195–202, June 1987.

[52] M. Nemani and F. N. Najm. "High-Level Area and Power Estimation for VLSI circuits". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 18(6):290–298, June 1999.

[53] M. Nemani and F. N. Najm. "Towards a High-Level Power Estimation Capability". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 15(6):588–598, June 1996.

[54] A. K. Murugavel, N. Ranganathan, R. Chandramouli, and S. Chavali. "Least-Square Estimation of Average Power in Digital CMOS Circuits". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(1):55–58, February 2002.

[55] A. K. Murugavel and N. Ranganathan. "Petri Net Modeling of Gate and Interconnect Delays for Power Estimation". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(5):921–927, October 2003.

[56] S. Bhanja and N. Ranganathan. "Switching activity estimation of VLSI circuits using Bayesian networks". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(4):558–567, August 2003.

[57] M.Kandemir L.Li, N.Vijaykrishnan and M.J.Irwin. "A Crosstalk Aware Interconnect with Variable Cycle Transmission". In *Design Automation and Test in Europe*, pages 102–107, 2004.

[58] K. T. Tang and E. G. Friedman. "Peak Crosstalk Noise Estimation in CMOS VLSI Circuits". In *Proceedings of International Conference on Electronic Circuits, Systems*, pages 1539–1542, 1999.

[59] L. H. Chen and M. M. Sadowska. "Aggressor Alignment for Worst-Case Crosstalk Noise". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 20(5):612–621, May 2001.

[60] S. Muddu A. B. Kahng and D. Vidhani. "Noise and delay uncertainty studies for coupled RC interconnects ". In *Proceedings of 12th Annual IEEE Intl. ASIC/SOC Conference*, 1999.

[61] W. Chen, S. K. Gupta, and M. A. Breuer. "Analytic models for crosstalk delay and pulse analysis under non-ideal inputs". In *Proceedings of IEEE Intl. Test Conference*, pages 809–818, 1997.

[62] P. Chen and K. Kuetzer. "Toward true crosstalk noise analysis". In *Proceedings of International Conference on Computer-Aided Design*, pages 132–137, 1999.

[63] M. Cuviello, S. Dey, X. Bai, and Y. Zhao. "Fault modeling and simulation for crosstalk in system-on-chip interconnects". In *Proceedings of International Conference on Computer-Aided Design*, pages 297–303, 1999.

[64] P. B. Sabet and P. Renault. "An Event-Driven Approach to Crosstalk Noise Analysis". In *Proceedings of the 36th Annual Symposium*, pages 319–326, 2003.

[65] P. Saxena and S. Gupta. "On Integrating Power and Signal Routing for Shield Count Minimization in Congested Regions". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(4):437–445, April 2003.

[66] E. S. Kuh T. Xue and D. Wang. "A post-global routing optimization optimization approach to crosstalk estimation and reduction". In *Proceedings of TECHCON*, September 1996.

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

[68] B. Victor and K. Kuetzer. "Bus Encoding to Prevent Crosstalk Delay". In *International Conference on Computer-Aided Design*, pages 57–63, 2001.

[69] L. Macchiarulo, E. Macii, and M. Poncino. "Wire Placement for Crosstalk Energy Minimization in Address Buses". In *Design Automation and Test in Europe*, 2002.

[70] P. Subrahmanya, R. Manimegalai, V. Kamakoti, and M. Mutyam. "A bus-encoding technique for power and crosstalk minimization". In *17th International Conference on VLSI Design*, pages 443–448, 2002.

[71] W. C. Cheng and M. Pedram. "Memory bus encoding for low power:a tutorial". In *International Symposium on Quality Electronic Design*, pages 199–204, 2001.

[72] C. G. Lyuh and T. Kim. "Low Power Bus Encoding with Crosstalk Delay Elimination". In *Proceedings of International Conference on Electronics Circuits and Systems*, pages 1539–1542, 1999.

[73] L. Benini, G. De Micheli, E. Macii, M. Poncino, S. Quer. "Power Optimization of core-based systems by address bus encoding". *IEEE Transactions on VLSI*, 6(4):554–562, Dec 1998.

[74] M. Becer and I. N. Hajj. "An Analytical Model for Delay and Crosstalk Estimation with Application to Decoupling". In *Proceedings of International Symposium on Quality Electronic Design*, pages 51–57, 2000.

[75] V.K. Rohatgi. *"An Introduction to Probability and Statistics"*. Wiley, 2001.

[76] J. R. Blum and J. I. Rosenblatt. *"Probability and Statistics"*. W. B. Saunders Company, 1972.

[77] S. Gupta and F. N. Najm. "Power Modeling for High-Level Power Estimation". *IEEE Transactions on VLSI*, 8(1):18–29, Feb 2000.

[78] H. Kriplani, F. N. Najm, and I. N. Hajj. "Pattern Independent Maximum Current Estimation in Power and Ground Buses of CMOS VLSI circuits: Algorithms, Signal Correlations, and Their Resolution". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 14(8):998–1012, Aug 1995.

[79] F. N. Najm, R. Burch, P. Yang, and I. N. Hajj. "Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 9(4):439–450, April 1990.

[80] F. N. Najm. "Transition Density: A new Measure of Activity in Digital Circuits". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 12(2):310–323, Feb 1993.

[81] E. S. Kuh T. Xue and D. Wang. "Post global routing crosstalk risk estimation and reduction". In *Proceedings of International Conference on Computer-Aided Design*, pages 302–309, November 1996.

[82] E. S. Kuh T. Xue and D. Wang. "Post global routing crosstalk synthesis". *IEEE Transactions on Computer-Aided Design*, 16(12):1418–1430, December 1997.

[83] D. Kirkpatrick and A. Sangiovanni-Vincentelli. "Techniques for crosstalk avoidance in the physical design of high-performance digital systems". In *Proceedings of International Conference on Computer-Aided Design*, pages 616–619, 1994.

[84] H. Chen and C. Wong. "Wiring and crosstalk avoidance in multi-chip module design". In *Proceedings of CICC*, pages 2861–2864, 1992.

[85] K. Chaudhary, A. Onozawa, and E. S. Kuh. "A spacing algorithm for performance enhancement and crosstalk reduction". In *Proceedings of International Conference on Computer-Aided Design*, pages 697–702, 1993.

[86] C-Y. Wang, and K. Roy. Performance-driven interconnect global routing. In *Proceedings on Sixth Great Lakes Symposium on VLSI*, pages 132–136, 1996.

[87] T. Gao and C. Liu. "Minimum crosstalk switchbox routing". In *Proceedings of International Conference on Computer-Aided Design*, pages 610–615, 1994.

[88] C. Gopalakrishnan and S. Katkoori. "A Fast Architectural Leakage Power Simulator for VHDL Structural Descriptions". In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI Tampa*, pages 211–212, February 2003.

[89] C. Gopalakrishnan and S. Katkoori. "Resource Allocation and Binding for Low Leakage Power". In *Proceedings of 16th International Conference on VLSI Design*, pages 297–302, January 2003.

[90] C. Gopalakrishnan and S. Katkoori. "KnapBind: An Area-Efficient Binding Algorithm for Low-leakage Datapaths". In *Proceedings of International Conference on Computer Design (ICCD)*, pages 430–435, October 2003.

[91] C. Gopalakrishnan and S. Katkoori. "Tabu Search Based Behavioral Synthesis of Low Leakage Datapaths". In *Proceedings of International Symposium on VLSI*, pages 260–261, 2004.

[92] P.G.Paulin and J.P.Knight. "Force-Directed Scheduling for the Behavioral Synthesis of ASICs". *IEEE Transactions on Computer-Aided Design of Circuits and Sustems*, 8(6):661–679, 1989.

[93] S. Gupta. "Force-directed scheduling for dynamic power optimization". *Master's thesis*, Spring 2002.

[94] G. De Micheli. *"Synthesis and Optimization of Digital Circuits"*. McGraw Hill Inc., 1994.

[95] S. H. Gerez. *"Algorithms for VLSI Design Automation"*. John Wiley and Sons, 1998.

[96] M. Sarrafzadeh and C. K. Wong. *"An Introduction to VLSI Physical Design"*. The McGraw-Hill Companies, Inc., 1996.

[97] C. Tseng and D. P. Siewiorek. "Facet : A Procedure for the Automated Synthesis of Digital Systems". In *20th ACM/IEEE Design Automation Conference*, pages 490–496, 1983.

ABOUT THE AUTHOR

Suvodeep Gupta received the Bachelor of Engineering (B.E.) degree in Electronics and Communication Engineering from Birla Institute of Technology, Mesra, India in 1999. He then joined the graduate program in the department of Computer Science and Engineering at the University of South Florida, Tampa, USA. He obtained his Masters degree in Computer Science and Engineering in 2002 and continued for a PhD. During the course of his graduate studies, he published multiple journal and conference papers. Besides research, he also taught several courses at the undergraduate level in the university. He is the recipient of the ACM-SIGDA scholarship for graduate students. His research interests include design automation, high-level synthesis, low-power VLSI design, and physical-level synthesis. He is a member of ACM and IEEE.