I, Thiagarajan Arumugam, hereby submit this work as part of the requirements for the degree of:
Master of Science
in:
Computer Engineering
It is entitled:
A heuristic approach for Capacitive Crosstalk
Avoidance during Post Global Routing Crosstalk
Synthesis for Deep Submicron Technologies

This work and its defense approved by:

Chair:
Dr. Harold W. Carter
Dr. Wen-Ben Jone
Dr. Carla Purdy
A heuristic approach for Capacitive Crosstalk Avoidance during Post Global Routing Crosstalk Synthesis for Deep Submicron Technologies

A thesis submitted to the
Division of Graduate Studies and Research of the
University of Cincinnati
in partial fulfillment of the
requirements for the degree of
MASTER OF SCIENCE
in the Department of
Electrical and Computer Engineering
of the college of Engineering
February, 2008

by

Thiagarajan Arumugam
B.E., Anna University, India 2005

Thesis Advisor and Committee Chair: Dr. Harold W. Carter
Abstract

With decreasing feature sizes, higher clock rates and increasing interconnect densities, crosstalk has become a major concern in Integrated Circuit (IC) design [1]. Crosstalk optimization is usually performed during placement, global routing and detailed routing. In order to achieve accuracy and flexibility, previous approaches proposed an intermediate step between global routing and detailed routing to perform the crosstalk synthesis.

In this work, we present techniques to improve upon the existing capacitive crosstalk aware router developed in [2] which uses a post global routing crosstalk synthesis step to perform crosstalk estimation and reduction. We propose a new approach to automate the process of identifying the amount of crosstalk sensitivity between the nets. Our objective function (noise margin) also significantly reduces the computation time. Finally, we arrive at a crosstalk-free net order for each region of the chip which serves as a good starting point for a detailed router.
In the loving memory of

Papuchi
Acknowledgements

I am extremely grateful to my advisor Dr. Hal Carter for providing me the opportunity to work under his guidance. His support and encouragement for the past two years have been unconditional and wholesome.

I would like to take this opportunity to thank Dr. Wen-Ben Jone and Dr. Carla Purdy for taking time off their busy schedules to be on the defense committee.

I would like to thank all my friends in UC for making my stay in Cincinnati a memorable one. Thanks to Arun, Babu, Haran, Harish, Romana, Sastha and Sucha in particular.

Special thanks to amma, appa, Arun, Raj, Sankar, Hema and Devi for always being there for me.
# Contents

1. Introduction ................................................. 1
   1.1 Motivation........................................... 2
   1.2 Problem Statement................................... 3
   1.3 Approach........................................... 4
   1.4 Research Contribution............................. 5
   1.5 Thesis Outline...................................... 5

2. Background ................................................ 7
   2.1 Crosstalk........................................... 8
      2.1.1 Capacitive Crosstalk........................... 10
      2.1.2 Capacitive Crosstalk Reduction.............. 13
   2.2 Physical Design Flow................................. 13
      2.2.1 Crosstalk Synthesis during Physical Design... 15
   2.3 Noise Margin....................................... 17
3. Existing Approaches ........................................ 19

3.1 Crosstalk Synthesis during Placement ................. 20

3.2 Crosstalk Synthesis during Global Routing ............. 21

3.3 Crosstalk Synthesized during Detailed Routing ....... 21

3.4 Post Global Routing Crosstalk Synthesis .............. 22

3.4.1 Crosstalk Risk Estimation ............................ 25

3.4.1.1 Crosstalk Risk Representation .................. 25

3.4.1.2 Crosstalk Violations .............................. 27

3.4.1.3 Graph-Based Crosstalk Risk Representation ........ 27

3.4.1.3.1 Crosstalk Risk Graph ....................... 27

3.4.1.3.2 Constrained Simple Path Sub-graph ............ 28

3.4.2 Crosstalk Risk Reduction ............................ 30

3.4.2.1 Crosstalk Risk Bound Partitioning ............... 31

3.4.2.2 Crosstalk Reduction .............................. 31

3.4.2.2.1 Shielding ....................................... 31

3.4.2.2.2 Rip Up / Re-route / Global Route Re-adjustment .... 32

3.4.3 Crosstalk-Free Net Order ............................ 32

3.4.4 Detailed Routing ....................................... 32
3.5 Recent Works in Post Global Routing Crosstalk Optimization .................................................. 33

4. Implementation .......................................................... 34

4.1 Post Global Routing Crosstalk Optimization ............... 36

4.1.1 Crosstalk Risk Estimation .............................. 36

4.1.1.1 Determination of Sensitivity between Nets .... 37

4.1.1.2 Risk Tolerance Bound .............. 41

4.1.2 Crosstalk Risk Reduction ...................... 42

4.1.2.1 Shielding ...................................... 42

4.1.2.2 Empty Tracks ................................. 43

4.1.2.3 Global Routes Re – Adjustment .... 43

4.2 Detailed Routing .................................................. 44

4.3 An Improved approach ........................................ 45

5. Results and Conclusion ........................................... 47

5.1 Results .................................................................. 48

5.1.1 Crosstalk Reduction ................................. 51

5.1.1.1 Total Crosstalk Value ....................... 51

5.1.1.2 Average Crosstalk Value ................. 54
List of Figures

Figure 2.1: A simple capacitive charge sharing model......................... 10
Figure 2.2: Equivalent circuit for the simple capacitive charge sharing
model................................................................. 11
Figure 2.3: Single victim line subjected to crosstalk from two adjacent
aggressors.............................................................. 12
Figure 2.4: The typical ASIC design flow.................................. 14
Figure 2.5: Various stages in the physical design flow to perform crosstalk
synthesis............................................................... 15
Figure 2.6: Noise margin levels................................................. 17
Figure 2.7: Noise margin calculation........................................... 18
Figure 3.1: Block diagram of the post global routing crosstalk optimization
(PGRCO) algorithm..................................................... 23
Figure 3.2: The steps in the PGRCO algorithm.............................. 24
Figure 3.3: The crosstalk risk estimation and reduction process.......... 25
Figure 3.4: Construction of a CRG_{csp} (e) from a CRG (e)........... 29
Figure 4.1: Block diagram of the overall PGRCO system................ 35
Figure 4.2: Various steps in the post global routing crosstalk optimization
process............................................................... 36
Figure 4.3: Simple charge sharing model.................................... 38
Figure 4.4: Charge sharing model extended to include wire resistance... 39
Figure 5.1: Graph showing the % TCV reduction (for Nm = 0.5V) for various
circuit sizes compared to the TCV values obtained from a crosstalk
un-optimized approach……………………………………………….. 52

Figure 5.2: Graph showing the % TCV reduction (for Nm = 0.35V) for various
circuit sizes compared to the TCV values obtained from a crosstalk
un-optimized approach……………………………………………….. 53

Figure 5.3: Graph showing the % ACV reduction (for Nm = 0.5V) for various
circuit sizes compared to the ACV values obtained from a crosstalk
un-optimized approach……………………………………………….. 55

Figure 5.4: Graph showing the % increase in area for the two crosstalk
optimization approaches compared to an un-optimized circuit… 60

Figure A1: Pseudo-code of the Timberwolf3.2 placement algorithm…….. 72

Figure A2: Acceptance function……………………………………………… 72

Figure A3: Pseudo-code of the global router…………………………………… 78

Figure A4: Pseudo-code of the left edge algorithm based channel router….. 80
List of Tables

Table 5.1: Line equation and R-squared value for the plots in Figures 5.1 & 5.2 ......................................................... 53

Table 5.2: Line equation and R-squared value for the plot in Figure 5.3 ……… 55

Table 5.3: Maximum glitch value (in volts) for noise margin = 0.5V and VDD = 1V ......................................................... 57

Table 5.4: Maximum glitch value (in volts) for noise margin = 0.35V and VDD = 1V ......................................................... 57

Table 5.5: Track count for noise margin = 0.5V and VDD = 1V ……… 59

Table 5.6: Execution times for the two crosstalk optimization algorithms ……… 61

Table A1: \( \alpha \) value for different temperatures ......................................................... 75

Table B1: TCV (in VCS) for noise margin = 0.5V (Ideal logic) and VDD = 1V ......................................................... 82

Table B2: TCV (in VCS) for noise margin = 0.35V (Ideal logic) and VDD = 1V ......................................................... 82

Table B3: ACV (in volts) for noise margin = 0.5V (Ideal logic) and VDD = 1V ......................................................... 83

Table B4: ACV (in volts) for noise margin = 0.35V (Ideal logic) and VDD = 1V ......................................................... 83

Table B5: Area of the circuit (in Sq. Lambda) ......................................................... 84
Chapter 1

Introduction

This thesis documents the research conducted to investigate existing approaches to crosstalk optimization during the physical design stages of the ASIC design flow. Typically, crosstalk is considered during the placement, global routing and detailed routing phases of the physical design stage. Recent works in this area have proposed a slight deviation from this conventional method to achieve superior results. Specifically, this thesis is targeted at crosstalk synthesis during an intermediate stage between global routing and detailed routing. We propose a heuristic approach to improve upon an existing crosstalk synthesis algorithm to achieve accuracy and speed in capacitive crosstalk estimation and reduction during a post-global routing step.
1.1 Motivation

In deep sub-micron processes, the width of the metal used for interconnection of the various gates in the circuit is continuously being decreased to control the die size whereas its height is being increased to keep the metal resistance low. Furthermore, the wire lengths are getting longer and longer. Due to the wires being longer, taller and thinner, the area for which the wires are running parallel to each other increases. Thus, the wire to wire capacitance has increased considerably. The wire capacitance is made up of two components, namely the wire to substrate capacitance and the wire to wire (coupling) capacitance. As more and more interconnect layers are accommodated on the chip, the distance from the higher metal layers to the substrate increases, thereby decreasing the substrate capacitance component of the total net capacitance. Thus capacitive coupling has thus become the dominant contributor to interconnect noise and delay. In fact, coupling capacitance between wires can account for 70% of the total wiring capacitance, even in 0.25µm processes [3]. Also, circuits are operated in high frequencies to achieve faster response. Speed is of vital importance in modern day processors and hence clock frequencies tend to be in the Giga hertz range. At higher clock frequencies, inductive crosstalk comes into the picture.

The supply voltage is scaled or reduced as the technology scales down to the deep sub-micron level. Due to the voltage scaling, the nets have become more susceptible to external noise and thus crosstalk has become an issue for all clock and signal wires. A new crosstalk metric has been introduced in the 2005 edition of the International
Technology Roadmap for Semiconductors [4] for Metal 1, intermediate and global wires. The metric calculates the line length where 25% of the switching voltage is induced on a minimum pitch victim wire. This critical line length for a minimum global wire in 2020 is less than 30% of the line length in 2005 [4].

To summarize, as feature sizes shrink to the nanometer scale (and due to the subsequent increase in the interconnect density) and circuits operate in high clock frequencies (in high GHz), crosstalk becomes a critical factor affecting the performance of the circuit. Coupling can cause noise leading to spurious logic transitions as well as change in delay (both increase and decrease in delay). Hence estimation and reduction of crosstalk noise is one the most important tasks that need to be performed during the design flow.

### 1.2 Problem Statement

Crosstalk profoundly affects the circuit performance in very deep submicron (VDSM) technology [1]. Capacitive and Inductive coupling are the two major sources of Crosstalk. Since Inductive Crosstalk becomes predominant only in circuits operating at high GHz clock frequencies in the high GHz range, we restrict our focus to capacitive coupling in this work. Crosstalk be reduced / controlled by following good layout design practices. As mentioned earlier, crosstalk synthesis can be performed during the various stages of the layout design process. In this work we improve upon an existing crosstalk-aware router which estimates and reduces capacitive crosstalk during a post global routing
optimization step. The two major inputs to the algorithm are the sensitivity of the nets and the risk tolerance bound of a net. We propose a model to automate the measurement of crosstalk sensitivity among the nets. Also, we propose noise margin as the new objective function to serve as the risk tolerance bound so as to effectively perform crosstalk minimization. This information is then used to arrive at a crosstalk free net order which will serve as a good starting point for a detailed router. Specifically this thesis reports on efforts to achieve the following objectives

- Develop a heuristic approach to improve upon the existing post global routing crosstalk optimization algorithm proposed in [2] which estimates and reduces crosstalk.
- To determine a net order for each routing region of the chip such that the region is crosstalk-risk free.

### 1.3 Approach

In this work, we propose a new approach to automate the measurement of crosstalk sensitivity for evaluating the crosstalk status of every net and use a new objective function as the crosstalk metric to improve upon an existing post global routing crosstalk synthesis algorithm developed in [2]. Also, we implement some algorithmic changes in order to speed up the optimization process to arrive at a crosstalk free net order. This net order is then used to aid the channel routing algorithm to achieve the final routing solution. The overall crosstalk value of the chip is then calculated and compared with the
results produced using the existing algorithm. Also, the results are compared with a
crosstalk un-aware routing scheme to show the effectiveness of our approach. The trade-
offs in using the crosstalk optimization algorithms as compared to the conventional
routing algorithm, namely the area and execution times are reported.

1.4 Research Contribution

In this work, the following significant contributions have been made in the development
of the algorithm for capacitive crosstalk aware routing.

- Automation of the measurement of the crosstalk sensitivity between every pair of
  nets.

- Algorithmic improvements / modifications to the existing post global routing
crosstalk synthesis algorithm developed in [2] in order to reduce the execution
time and increase the accuracy.

- Proposed a new objective function to use as a metric to account for the crosstalk
  in the chip.

1.5 Thesis Outline

This thesis is organized as follows:

- Chapter one on **Introduction** provides the motivation, problem statement, the
  research approach and the contribution of this work.
• Chapter two on **Background** describes the basics of crosstalk, physical design and crosstalk synthesis during physical design.

• Chapter three on **Existing Approach** gives an overview of existing approaches to crosstalk synthesis during placement, global routing and detailed routing. We also present in detail the pot global routing crosstalk optimization algorithm proposed in [2].

• Chapter four on **Implementation** explains the actual implementation of the algorithm in the post global routing stage with the improvements we propose in this thesis.

• Chapter five on **Results and Conclusions** puts forth the results obtained from the algorithm in chapter 4 and the comparison of our work with the algorithm proposed in [2] and also the comparison of our work with a crosstalk unaware placement and routing algorithm. It also presents the conclusions of the research.

• Chapter six on **Future work** discusses about the scope for future work in this area and how the proposed approach can be further improved to achieve better results.
Chapter 2

Background

The process of estimating and reducing the crosstalk in a chip is termed as crosstalk synthesis. Before venturing into more specific details about how crosstalk synthesis is performed, it is important to introduce the various terminologies and concepts used throughout this thesis. We discuss briefly about crosstalk, its effects and ways to minimize it. Also, we explain the various stages of physical design in detail to give a better understanding of how crosstalk synthesis can be performed in these stages. The objective function in our work is to achieve a routing solution such that crosstalk risk is lesser than a bound - noise margin. Hence, we discuss briefly about noise margin and ways to measure the noise margin of a gate.
2.1 Crosstalk

An unwanted coupling between two neighboring signal wires introduces an interference which is called crosstalk. In other words, the voltage change in one signal wire (Aggressor) will couple to an adjacent wire (Victim) thereby inducing noise. Coupling between these signal lines can cause logic failures and timing degradation in the circuit. These problems become increasingly severe in deep submicron dimensions where the wire-to-wire capacitance dominates the total capacitance value. Hence estimation and reduction of noise becomes extremely important for modern IC chips.

Crosstalk can be caused by two kinds of coupling:

- **Capacitive Coupling:** Two wires routed adjacent to each other will have coupling capacitance (like any parallel plate capacitor) between them. The capacitance value can be obtained by using the formula shown in equation (2.1).

\[
C = \frac{\varepsilon A}{d}
\]

Where \( C \rightarrow \) Capacitance in Farad

\( \varepsilon \rightarrow \) Permittivity of the dielectric medium between the parallel plates

\( A \rightarrow \) Area of the parallel plates in Sq. meters

\( D \rightarrow \) Distance between the two parallel plates (adjacent nets) in meters

Hence the capacitance is directly proportional to the area of the metal wire and is inversely proportional to the distance of separation.
With advancement in IC technology, device dimensions are reaching the nanometer range. In order to control the die size, the width of the metal is continuously being decreased whereas its height is being increased to keep the resistance low. Furthermore, the wire lengths are increasing. Due to these long and thin wires, wire to wire capacitance has increased considerably. Capacitive coupling is the dominant contributor to the total capacitance in deep submicron technologies. The problems caused by coupling are explained in the pages that follow.

- **Inductive Coupling:** The wires are likely to have some self inductance and some mutual inductance. A current flowing on one of the conductors induces a voltage on the other conductor. Inductance causes oscillations in the output waveform. The effect is prominent for circuits operating in high clock frequency.

Crosstalk noise between neighboring signal wires causes two major problems that affect the operational integrity of IC designs [5]:

- **Crosstalk delay** changes the signal propagation on some of the nets, reducing the achievable clock speed.

- **Crosstalk glitch** causes voltage spikes on some nets, resulting in false logic states being captured in the registers.

This work concentrates mainly on the glitch effects of crosstalk caused by capacitive coupling.
2.1.1 Capacitive Crosstalk

On-chip wires have significant capacitance to adjacent wires on same layer and on adjacent layers. A signal voltage propagating on one of the conductors (Aggressor) will thus couple current onto the adjacent conductor (Victim) causing a coupled noise voltage. A victim line can have multiple aggressor lines. Let us denote the aggressor as A and the victim as V. Let $V_a$ be the aggressor voltage and $V_v$ be the voltage induced on the victim wire by the aggressor wire.

A change in voltage in the aggressor line will capacitively couple with the victim wire and will cause a voltage change in the victim. If the victim net is floating, the signal will not be restored to its original value, whereas if the victim net is also driven the voltage will be restored to its original value within a time constant. A charge sharing model is commonly used for modeling the crosstalk noise. Consider Figure 2.1. $C_c$ is the coupling capacitance between the aggressor (A) and victim (V). $C_o$ is the capacitance-to-ground of the victim line.

![Figure 2.1: A simple capacitive charge sharing model](image)
The circuit in Figure 2.2 is equivalent to the one in Figure 2.1. $V_a$ and $V_v$ are the aggressor and victim voltages respectively. $S$ is a switch which when closed will couple the two nets.

![Figure 2.2: Equivalent circuit for the simple capacitive charge sharing model [20]](image)

The change in victim voltage can be measured using the charge sharing equation (assuming the victim line is not in transition)

$$V_v = V_a \left( \frac{C_c}{C_c + C_o} \right) \quad (2.2)$$

Note that even though this model is simple to calculate, it is not accurate as it ignores victim and aggressor resistances completely. In modern ICs, the wires span the entire area of the chip and hence the wire resistance can no longer be ignored. A crosstalk model that we use in this work which includes the resistance effects will be presented in Chapter 4. As mentioned earlier in this chapter, a victim line can have multiple aggressors. For capacitive crosstalk, we usually consider the effect of the two adjacent aggressors (on either side of the victim) and ignore the effect due to other aggressors as
they are negligible. Consider the case of a single victim line subjected to crosstalk by two adjacent aggressor lines (A1 and A2) as shown in Figure 2.3.

![Diagram of A1, V, and A2 with coupling capacitance Cc](image)

Figure 2.3: Single victim line subjected to crosstalk from two adjacent aggressors

The value of the coupling capacitance in this case will depend on the direction of the signal swings on the aggressors and the victim. If the victim voltage is constant (i.e. no signal transition) and the aggressor voltages are simultaneously in transition in the same direction, the total capacitance seen by the victim is 2Cc. If the victim line voltage is in transition and the aggressor voltages swing in the opposite direction to the victim voltage’s swing, the capacitance due to each of aggressor will double because of miller effect* and the effective capacitance seen by the victim line is 4Cc.

The above analysis holds good for measurement of the glitch amplitude caused by coupling capacitance. The delay of the victim line has the following relationship to the signal transitions. [6, 7]

- If two coupled nets switch in the same direction, they can have lower signal delay.
- If they switch in the opposite directions, they can have longer delay.

* Miller Effect: A capacitor experiencing identical but opposite voltage swings at both its terminals can be replaced by a capacitor to ground, whose value is two times the original value [6].
2.1.2 Capacitive Crosstalk Reduction

The capacitive crosstalk between two wires is proportional to the coupling capacitance between the wires, which in turn is proportional to their coupling length (the total length of their overlapping segments) and inversely proportional to their separating distance as evident from Equation (2.1). Thus, an increase in the distance of separation between the wires or a decrease in the coupling length will help reduce the coupling capacitance in turn minimizing crosstalk. It is common practice to do crosstalk optimization during the layout design stage of the ASIC design flow. The various steps in the physical design flow are explained in Section 2.2.

2.2 Physical Design Flow

The typical ASIC design flow diagram is shown in Figure 2.4. It takes as its input the circuit specifications from the user/consumer. A behavioral model of the circuit is then created using hardware description languages like VHDL or Verilog to verify the functionality of the circuit. A RTL level synthesizable model is then created and subjected to synthesis. The synthesis takes as its inputs typical design constraints such as area, power and timing. The resulting net-list from synthesis is then verified for functionality (by simulation) and timing (by static timing analysis). The flow then reaches the physical design stage which starts with the clock tree synthesis. Since clock is the most sensitive net in the circuit (i.e. it has maximum switching), clock nets are routed
Figure 2.4: The typical ASIC design flow
ahead of the signal nets. Floor planning is then performed to determine possible locations for the cells (logic gates) in the chip. During placement, all the cells are assigned to some part of the chip. In the global routing stage, the nets are assigned to particular routing channels of the chip. In the detailed routing phase, the nets are allotted to their respective tracks to finalize their routes.

The layout is then verified for design rule violations using a DRC checker. A layout VS schematic tool (LVS) is then used to compare the layout with the schematic to validate the results. A GDS / CIF file is then created and sent for fabrication.

2.2.1 Crosstalk Synthesis during Physical Design

As mentioned in section 2.1, crosstalk reduction can be achieved by proper layout design practices. Crosstalk synthesis (i.e. estimation and reduction of crosstalk) can be done at various stages of the physical design flow as shown in the Figure 2.5.

![Crosstalk Synthesis Diagram]

Figure 2.5: Various stages in the physical design flow to perform crosstalk synthesis
Crosstalk synthesis during placement is inaccurate because of the unavailability of any routing information at that stage. Moreover crosstalk noise is routing dependent since the coupling-capacitances between nets are determined by the routes of interconnect wires on the chip. Therefore, it is most appropriate to address crosstalk risk estimation and reduction during the routing process [2].

The crosstalk synthesis during global routing is also not accurate because the actual route/track information and orientation of the nets are not known at this stage. Global routing assigns routes to its channels but does not provide any information about the adjacency of nets. The adjacency information becomes available only during the detailed routing phase.

The crosstalk synthesis during detailed routing is accurate but is not flexible because the detailed routing works on routing channels independently and hence cannot have the flexibility of considering the route of the nets globally among all the routing regions in the chip. For instance, let us assume that a channel has five nets and we find that one of the nets is sensitive to all the four other nets in this channel. Hence this net is susceptible to crosstalk when routed in this channel. Ideally, we would want to remove the net in question from this channel and re-route it in another routing channel. But since we are working in the detailed routing phase, we don’t have the flexibility to re-route this net in another region of the chip. Later works in this area have proposed new methods to make use of the flexibility and accuracy of all the above methods to provide better solutions.
2.3 Noise Margin

Noise margin is the measure of sensitivity of a gate to noise. For a gate to be robust and insensitive to noise disturbances, it is essential that the ‘0’ and ‘1’ intervals be as large as possible [6]. These levels are denoted by the noise margin low (NM_L) and noise margin high (NM_H) values as shown in Figure 2.6. The noise margins should be greater than 0 for a digital gate to be functional and preferably be as high as possible. The region between NM_H and NM_L is called the undefined region, uncertainty region or the transition width. This width should be as small as possible and should be avoided by the steady state signals if proper circuit operation is to be ensured.

![Diagram of noise margin levels](image)

Figure 2.6: Noise margin levels [6]

The noise margin high and noise margin low are given by the relation

\[
NM_H = V_{IL} - V_{OL} \quad NM_L = V_{OH} - V_{IH}
\]  

(2.3)
An ideal inverter will have a zero uncertainty level, i.e. the high and low noise margins will be equal to half the swing (supply voltage). Thus, for an ideal inverter \( \text{NM}_H = \text{NM}_L = \frac{V_{DD}}{2} \), where \( V_{DD} \) is the supply voltage. The noise margin values can be obtained from the voltage transfer characteristics (VTC) of a gate using the points where the gain of the VTC equals -1 as shown in Figure 2.7.

![Figure 2.7: Noise margin calculation [6]](image)

The noise margin values can also be obtained from the standard cell library if available. In this research work, we use the noise margin of the gates as a measure to gauge the amount of crosstalk that a net (that drives these gates) can be subjected to without causing abnormal behavior of the circuit. If the crosstalk noise amplitude of a net exceeds the noise margin levels of the gate it is driving, then optimization of crosstalk is needed in order to avoid spurious logic transitions in the gate. A detailed description of how this is done is presented in Chapter 4.
Chapter 3

Existing Approaches to Crosstalk Synthesis

Crosstalk synthesis (i.e. estimation and reduction of crosstalk) can be performed at various stages of physical design - placement, global routing and detailed routing. In this chapter, we give a brief description of previous works in this area. Also, the various disadvantages of synthesizing crosstalk during these stages were mentioned in Chapter 2. Recent works have proposed ways to avoid these disadvantages by introducing new approaches to crosstalk synthesis. One such approach developed in [2] is presented in detail in this chapter. Our research work is an extension of the algorithm developed in [2].
3.1 Crosstalk Synthesis during Placement

Placement is the stage where all the cells in the chip are assigned to particular locations in the chip such that the cost function is reduced. Typical cost functions to placement are area, aspect ratio and net length. As technology scaled down, the device sizes shrunk to nanometer scale and hence became more susceptible to noise. Thus new cost functions like crosstalk reduction were introduced in placement algorithms.

One such approach is presented in [8] where the placement algorithm estimates and optimizes coupling capacitance using a probabilistic model. They control the placement density to reduce the number of highly coupled regions. However, crosstalk is not only dependent on coupling capacitance but also on the driver resistance and wire resistance of the nets in deep submicron (DSM) processes. Hence a more accurate model is needed to represent the effects more accurately. [9] introduced a noise map to model crosstalk noise and then used it to guide an incremental placement algorithm to mitigate noise.

The algorithms proposed in the works mentioned above suffer from the fact that the crosstalk synthesis during placement has no information about routing. Moreover crosstalk noise is routing dependent since the coupling-capacitances between nets are determined by the routes of interconnect wires on the chip. Therefore, it is most appropriate to address crosstalk risk estimation and reduction during the routing process [2].
3.2 Crosstalk Synthesis during Global Routing

In the global routing stage, the nets are assigned to particular routing channels of the chip. No information about the tracks on which the nets are going to be placed is available at this stage. [10] presented a global routing algorithm based on Steiner tree formulation and the Lagrangian relaxation technique which tries to reduce the crosstalk noise during global routing. A global routing approach which includes RLC effects was proposed in [11] using simultaneous shield insertion and net ordering.

These crosstalk synthesis approaches suffer from a similar problem as the ones presented in section 3.1. The actual positions of the nets in their tracks are not available in this stage making it impossible for a crosstalk optimization algorithm to get adjacency information. Since coupling crosstalk affects nets which are adjacent to each other, the adjacency information is vital in arriving at an optimization solution.

3.3 Crosstalk Synthesis during Detailed Routing

In the previous section, we mentioned that the adjacency information is important for performing an accurate crosstalk optimization. During the detailed routing phase, the nets are allotted to their respective tracks to finalize their routes. This gives us the needed information about adjacency thus making crosstalk synthesis during detailed routing more accurate than the other two approaches. Several works have concentrated on
treating crosstalk during the detailed routing stage. [12], [13] and [14] are three such examples.

Even though it is accurate, the detailed routing crosstalk optimization process is not flexible because it works on routing channels one at a time and hence cannot have the flexibility to correct the failure if a global optimization is required. For instance, let us assume that a channel has five nets and we find that one of the nets is sensitive to all the four other nets in this channel. Hence this net is susceptible to crosstalk when routed in this channel. Ideally, we would want to remove the net in question from this channel and re-route it in another routing channel. But since we are working in the detailed routing phase, we don’t have the flexibility to re-route this net in another region of the chip.

### 3.4 Post Global Routing Crosstalk Synthesis

In order to avoid the various drawbacks of the conventional crosstalk synthesis explained in the previous sections, recent works have proposed new approaches to crosstalk synthesis. In order to avoid the various drawbacks of the conventional crosstalk synthesis explained in the previous sections, recent works have proposed new approaches to crosstalk synthesis. One solution is to perform crosstalk optimization during an intermediate stage between global routing and detailed routing. This stage is named the post global routing stage.
One such approach developed in [2] works in a post global routing step to perform crosstalk synthesis. Our research work is an extension of the algorithm developed in [2]. We give a brief description of the algorithm in this section.

![Block diagram of the post global routing crosstalk optimization algorithm](image)

Figure 3.1: Block diagram of the post global routing crosstalk optimization algorithm [2]

The block diagram of the post global routing crosstalk optimization algorithm presented in [2] is shown in Figure 3.1. The algorithm takes as its input a feasible global routing solution of the chip, sensitivities and risk tolerance bounds of nets and produces a risk-free routing solution in which all regions of the chip are free of crosstalk violations. The authors assume that the sensitivity information and the risk tolerance bounds of nets can be extracted using temporal and functional analysis or can be specified by the user. We will later show that this information can be obtained using the model we propose which will improve the accuracy of the approach. The algorithm consists of two major steps
• Region based crosstalk risk estimation
• Crosstalk risk reduction at the global routing level

In part one, crosstalk risk graphs are used to represent the current crosstalk situation of the chip and the crosstalk risk of each region is determined. The number of shields required for a crosstalk free routing solution is then estimated.

In part two, risk tolerance bound of each net is partitioned among the routing regions. Shielding is used to reduce the positive risks in various regions of the chip. If enough shields are not available in a region, the global route of the net is readjusted to accommodate the net in another region where it is not subjected to crosstalk noise.

Finally, if net rip-up and re-route is not successful in achieving the intended risk free solution, the global routing step is re-processed to obtain a new global routing solution which is then subjected to all the above steps. Figure 3.2 shows the various steps in the algorithm.

![Figure 3.2: The steps in the PGRCO algorithm [8]](image-url)
As shown in the Figure 3.2, the region based crosstalk risk estimation first constructs a crosstalk risk graph for each routing region representing its current crosstalk situation based on the initial risk partition bounds. The impact of bound changes is analyzed and the current partitions of the bounds are adjusted via integer linear programming. If positive risks still exist, global routes adjustment is applied. Nets whose removal leads to a decrease in the overall crosstalk in the region are ripped up and re-routed with minimum cost alternatives. These steps iterate until a risk-free global routing solution is obtained. The crosstalk risk estimation and reduction process is shown in Figure 3.3.

![Figure 3.3: The crosstalk risk estimation and reduction process [8]](image)

### 3.4.1 Part One: Crosstalk Risk Estimation

#### 3.4.1.1 Crosstalk Risk Representation

It is assumed that crosstalk noise exits only between nets routed in adjacent tracks in a region. Noise between nets one or more tracks apart is ignored. Noise between some adjacent net pairs may not affect the proper functioning of the circuit due to logical and
temporal isolations [15]. For instance, two nets that are not active at the same time may be immune from the noise spike. This information is represented as the sensitivity of nets, say $S_{ij}$. Sensitivity information as well as the risk tolerance bounds are assumed to be extracted using temporal and functional analysis or specified by the user.

$S_{ij} \in (0,1)$ and $S_{ij} = 1$ implies that nets $i, j$ are subject to crosstalk and $S_{ij} = 0$ implies they are crosstalk-safe when routed in adjacent tracks. If $N$ denotes the set of nets, a net is said to be a sensitive net (i.e. it belongs to the sensitive net set $N_s$) if it is sensitive to at least one other net on the chip. Let $\text{noise}(i,j)$ represent the noise between adjacent nets $i$ and $j$. Since the coupling capacitance between $i$ and $j$ is proportional to the coupling length $\text{len}(i,j)$ (i.e. the length over which the two nets $i$ and $j$ run parallel to each other), $\text{noise}(i,j)$ can be expressed as

$$\text{noise}(i,j) = S_{ij} \times \text{len}(i,j) \quad (3.1)$$

If $\text{Bound}(i)$ represents the risk tolerance bound of a sensitive net $i \in N_s$ (i.e. the maximum crosstalk the net $i$ can tolerate without affecting the functionality of the circuit), the condition below makes sure that net $i$ is safe from crosstalk.

$$\sum_{e \in \text{route}(i)} \sum_{j \in \text{Adj}(i,e)} \text{noise}(i,j,e) = \sum_{e \in \text{route}(i)} \sum_{j \in \text{Adj}(i,e)} S_{ij} \times \text{len}(i,j,e) < \text{Bound}(i) \quad (3.2)$$

Where $\text{Adj}(i,e)$ is the set of nets adjacent to net $i$ in the routing region $e$ and $\text{route}(i)$ represents the various regions through which $i$ is routed. Since the crosstalk noise at net $i$ comes from all regions on its route according to [16], the $\text{Bound}(i)$ was partitioned accordingly among $\text{route}(i)$ for region-based crosstalk estimation and optimization in [2].
Hence

\[ \text{Bound}(i) = \sum_{e \in \text{route}(i)} \text{Bound}(i, e) \]  \hspace{1cm} (3.3)

The bound partitioning technique used in [2] is explained later in this chapter.

3.4.1.2 Crosstalk Violations

During global routing, each net occupies an entire track and hence is adjacent to no more than two nets. Crosstalk violation may occur at net \(i\) in region \(e\) only during the following two cases:

- Case 1: The noise from one of \(i\)’s adjacent nets in \(e\) violates its risk tolerance bound.
- Case 2: Net is safe under Case 1, but the summation of noises from both of \(i\)’s adjacent nets in \(e\) violates its bound.

3.4.1.3 Graph-based Crosstalk Risk Representation

3.4.1.3.1 Crosstalk Risk Graph

A crosstalk risk graph \(\text{CRG}(e) = (N_s(e), E_s(e))\) is constructed for each region \(e\) to represent its current crosstalk situation. Each node \(i\) in \(\text{CRG}(e)\) corresponds to a sensitive net routed in \(e\), its weight representing the partitioned risk tolerance bound of net \(i\) in \(e\), i.e. \(\text{Bound}(i, e)\). The weight of each edge between the nodes \(i, j\) represent the potential crosstalk noise between nets \(i\) and \(j\) in region \(e\) if they are routed in adjacent tracks. An
edge $\text{Edge}(i,j) \in \text{Es}(e)$ if and only if $\text{noise}(i,j,e)$ is less than the partitioned risk tolerance bounds of both nets $i$ and $j$ in $e$ as shown in Equation (3.4).

$$\text{noise}(i,j,e) < \text{Bound}(i,e) \text{ and } \text{noise}(i,j,e) < \text{Bound}(j,e) \quad (3.4)$$

Hence, each edge $(i,j)$ in $\text{CRG}(e)$ implies that nets $i,j$ can be routed in adjacent tracks without causing crosstalk violations at nets $i,j$. Hence crosstalk violation under Case 1 has been accounted for, but the fact that nets $j,k$ are compatible with net $i$ separately does not guarantee they can be placed adjacent to $i$ at the same time, since summation of noise from $j$ and $k$ can cause crosstalk violation at $i$ due to Case 2.

A Constrained Simple Path Subgraph ($\text{CRG}_{\text{csp}}(e)$) is used in [2] for this purpose. The section below explains the $\text{CRG}_{\text{csp}}(e)$ in detail.

### 3.4.1.3.2 Constrained Simple Path Sub-graph

The constrained simple path subgraph is represented as $\text{CRG}_{\text{csp}}(e) = (\text{Ns}(e),\text{Ep}(e))$ where $\text{Ep}(e)$ belongs to $\text{Es}(e)$. $\text{CRG}_{\text{csp}}(e)$ contains simple path segments only, i.e. degree $\leq 2$ holds for every node $i$ in $\text{CRG}_{\text{csp}}(e)$. In addition, the weight of each node $i$ is greater than the summation of the weights of the two edges connecting to the node.

$$\text{noise}(i,j,e) + \text{noise}(i,k,e) < \text{Bound}(i,e) \quad (3.5)$$
This condition ensures that violations under Case 2 have been avoided. Many CRG\textsubscript{csp}(e)’s may exist for a CRG(e). The focus is on constructing a CRG\textsubscript{csp}(e) having the most number of edges (which will ensure that more number of nets can be placed adjacent to each other giving more flexibility to the router).

An example CRG(e) and a CRG\textsubscript{csp}(e) constructed from that CRG(e) is shown in Figure 3.4. The diagram to the left shows the CRG(e) for a region and the figure to the right shows one of the CRG\textsubscript{csp}(e)’s that can be obtained from that CRG(e).

Figure 3.4: Construction of a CRG\textsubscript{csp}(e) from a CRG(e) [2]

Once the CRG\textsubscript{csp}(e) is formed, the algorithm tries to find a Hamiltonian path for the graph. A Hamiltonian path in a graph is defined as a special simple path segment that visits every node in the graph only once. Hence, a risk free routing solution for the region exists if a Hamiltonian path exists in one of the CRG\textsubscript{csp}(e)’s of the CRG(e).
If multiple trees (simple path segments) exist in a $\text{CRG}_{csp}(e)$ of region $e$, the nets corresponding to the root/leaf nodes of these path segments cannot be routed together in adjacent tracks. Hence, to form a Hamiltonian path for the $\text{CRG}_{csp}(e)$, we insert shields between the end nodes. The number of shields required and the number of shields available can be estimated by using analytical techniques as explained in [2].

The difference between the shields needed and the shields available will give the value of $\text{risk}(e)$ of region $e$. If $\text{risk}(e) \leq 0$, it implies there are more than enough shields in the region. If $\text{risk}(e)$ is positive, it gives the value of the extra shields that are needed for the region $e$ to be risk free. In such cases, the crosstalk reduction step explained in the subsequent sections will help in minimizing the number of shields required.

The risk tolerance bound partition and update step is used to accurately update the bounds to account for the actual crosstalk risk of each region.

### 3.4.2 Part Two: Crosstalk Risk Reduction

Once the crosstalk risk of each region is estimated and the regions with positive risks are identified, the crosstalk risk reduction step proceeds by eliminating the positive risks regions on the chip so that every routing region has a risk-free global routing solution.
3.4.2.1 Crosstalk Risk Tolerance Bound Partitioning

As mentioned in the closing paragraphs of the previous section, the positive risks of the chip are accurately estimated and reduced during the bound partition and update step. The objective of the crosstalk risk bound partitioning step is to “Partition the risk tolerance bound of each net among its routing regions to reflect their crosstalk situations so that the total positive risk of regions is minimized and an accurate estimation of regions’ risks can be obtained” [2]. A two-phase integer linear programming (ILP) formulation is used in [2] to perform the adjustments in bound.

In our work, we have eliminated the need to do the risk bound partitioning as explained in Chapter 4.

3.4.2.2 Crosstalk Reduction by Shielding, Track Assignment and Global Route Re-Adjustment

3.4.2.2.1 Shielding

In the CRG, two disjoint trees are connected by a shield to form a Hamiltonian path. This means that if two nets in a region are sensitive to each other and the crosstalk noise between them is more than the risk tolerance bounds of the gates the nets are driving, then those two nets cannot be placed in adjacent tracks. To avoid this, we can insert shields between the two nets. A shield can essentially be any one of the following:
• Power / Ground rail
• A non-sensitive net
• An empty track

3.4.2.2 Rip Up and Re-Route / Global Route Re-Adjustment

If enough shields are not available in a region, the net under consideration is ripped up and re-routed with minimum cost alternatives in any other region in the chip. The risk tolerance bounds are then updated accordingly. The flexibility to operate in different routing regions is attributed to the fact that we are working in a post global routing processing step.

3.4.3 Crosstalk-free Net Order

The application of the algorithm explained in this chapter will result in a graph with a Hamiltonian path. Traversing the graph from the root to leaf will give a sequence of nets in a particular order. If the region is routed with nets in this order, the region is said to be crosstalk-free.

3.4.4 Detailed Routing

The crosstalk-free net order obtained in the previous sections will serve as a good starting point for any crosstalk aware detailed router [2]. The ordering of nets obtained doesn’t
guarantee a zero crosstalk value, but the crosstalk values obtained will be below the bounds of each net thereby avoiding any spurious and undesirable effects.

### 3.5 Recent Works in Post Global Routing Crosstalk Optimization

There have been a few recent works that concentrate on the post global routing phase of crosstalk optimization. For instance, [21] proposed a post global routing RLC budgeting algorithm based on Linear Programming (LP) and applied it to the shield insertion and net ordering (SINO) proposed by [22] at full-chip level. The approach in [21] reported a 7% decrease in routing area and reduced run time compared to the best available existing approach. In [21], the authors showed that the approach proposed in [2] does not hold when inductive crosstalk comes into the picture. This is because, the assumption in [2] that crosstalk only exists between adjacent nets no longer holds for inductive crosstalk. In [11], an extended global routing algorithm with crosstalk constraints was presented. It uses simultaneous shield insertion and net ordering with track assignment to achieve a global routing solution. The results reported in that work showed 18% reduction in congestion.

In this research work, we propose a shield insertion and net ordering approach similar to [2] with a new objective function that results in a 3% reduction in area and up to 44% reduction in run time compared to the work in [2].
Chapter 4

Implementation

The research reported in this thesis is an extension of the post global routing crosstalk optimization algorithm presented in [2]. We propose a method to automate the calculation of sensitivity of a net by providing an accurate model which incorporates the effects due to coupling capacitance and the wire resistance. We suggest a new metric, namely noise margin to serve as the upper bound for the crosstalk reduction. Also, our objective function avoids the use of the computation expensive step proposed in [2] thereby providing considerable speedup. A hierarchical block diagram of the overall system showing the steps involved and the algorithms used in achieving those steps is shown in Figure 4.1.
As shown in Figure 4.1, the PGRCO algorithm takes as its input a feasible global routing solution, net sensitivities and crosstalk bounds. A simulated annealing based standard cell placement and routing algorithm ‘Timberwolf3.2’ developed in [17] was used for performing the place and route. The Timberwolf3.2 algorithm and the various parameters used is explained in Appendix A. The algorithm proposed in [2] along with the improvements suggested in this research is then applied to arrive at a crosstalk-free routing order which is then applied to a detailed router (developed using the channel routing algorithm described in appendix A) to achieve the final routing solution of the chip.
4.1 Post Global Routing Crosstalk Optimization (PGRCO)

The global routing solution is subjected to the post global routing crosstalk optimization process. Figure 4.2 shows the various steps involved in the PGRCO.

![Diagram of PGRCO process]

Figure 4.2: Various steps in the post global routing crosstalk optimization process

As seen from Figure 4.2, the post global routing crosstalk optimization process involves two major steps:

4.1.1 Crosstalk Risk Estimation

As explained in Chapter 3, the crosstalk risk of the various regions on the chip is measured using the crosstalk risk graph (CRG) and the subsequent graph simplification techniques. Once the current crosstalk situation of the chip is estimated, the algorithm
proceeds by performing the crosstalk risk reduction step. The crosstalk risk estimation uses the sensitivity information to form the crosstalk risk graph. The procedure used in this research to determine the sensitivity between any two nets is explained in the section below. Sensitivity determination is one of the improvements of this work compared to [2]. Here we derive a model to automatically determine the value of sensitivity rather than assuming that it is available from the user as input.

4.1.1.1 Determination of Sensitivity between Nets

The crosstalk risk between any two nets is denoted by the sensitivity between these two nets when routed adjacent to each other. In [2], the authors assume that the sensitivity value is available as input from the user or calculated through functional simulation. Since functional simulation will require a prior knowledge of the circuit inputs and functionality, it is not always feasible for the designer to obtain the sensitivity information in this manner. Moreover, this will involve an additional time consuming step of simulating the circuit with test patterns to determine how each nets behaves. Also, the assumption that the sensitivity information can be obtained from the user is not realistic as this information depends on a number of parameters like net length and coupling capacitance which are not readily available to the user before actually performing the simulation to extract the parasitics. Hence, there is a need to determine this information during the design process to accurately estimate the sensitivity values. The model used to measure the sensitivity value is presented in the next section.
Sensitivity Model

The sensitivity between any pair of nets can be calculated using a simple charge sharing model as shown in Figure 4.3.

![Simple charge sharing model](image)

Using this model, we can calculate the change in voltage $\Delta V_{\text{victim}}$ on the victim net because of a change in the voltage $\Delta V_{\text{aggressor}}$ on the aggressor wire as given by Equation (4.1)

$$\Delta V_{\text{victim}} = \frac{C_{\text{adj}}}{C_{\text{gnd-v}} + C_{\text{adj}}} \Delta V_{\text{aggressor}}$$  \hspace{1cm} (4.1)

[18] presented a model that includes the effects of driver resistance. At the deep submicron dimensions, net lengths increase and hence interconnection length can no longer be ignored. Hence, wire resistance contributed by the longer interconnects need to be accounted for in the model presented above. Thus, resistances Re1 and Re2 are included in the charge sharing model (with driver resistances) as shown in Figure 4.4.
The value of $\Delta V_{\text{victim}}$ can now be calculated using the formula

$$\Delta V_{\text{victim}} = \frac{Cc}{C_{gv} + Cc} \times \frac{1}{1 + k} \times \Delta V_{\text{aggressor}}$$  \hspace{1cm} (4.2)$$

$$k = \frac{(Ra + Re1) (Cga+Cc)}{(Rv + Re2) (Cgv + Cc)}$$  \hspace{1cm} (4.3)$$

Where,

- $\Delta V_{\text{victim}}$ and $\Delta V_{\text{aggressor}}$ are the change in victim and aggressor voltages respectively.
- $Cc$ is the coupling capacitance
- $C_{gv}$ and $Cga$ are the capacitance to ground from the victim wire and the aggressor wire respectively
- $Re1$ and $Re2$ are the wire resistances of the victim wire and the aggressor wire respectively
The value of $\Delta V_{\text{victim}}$ is the measure of sensitivity between the nets. We calculate the sensitivity $S_{ij}$ for every pair of nets in the circuit and store it in a matrix $S$. [2] restricts the values for the sensitivity elements in the matrix $S$ to two. A ‘1’ in the matrix at the location $i, j$ indicates that nets $i$ and $j$ are sensitive. A ‘0’ will indicate they are not sensitive. They obtain the noise values by multiplying the length of the wire with the sensitivity value. In our work, we allow the sensitivity matrix to contain real numbers to indicate the amount to which they are sensitive. In real circuits, it is possible that a net $A$ is more sensitive to net $C$ than net $B$ to $C$. Hence it is more accurate to represent the sensitivities as real numbers rather than assuming them to be ‘1’ or ‘0’. This will make the algorithm more optimistic in its approach. Also, since we include the length parameter ($R_{e1}$, $R_{e2}$, $C_c$ depend on the length) in the formulae used to find the sensitivity, we avoid the need for multiplying the length to the sensitivity value separately.

Once $S$ is formed, the crosstalk risk graph is constructed for each region in the chip. In the crosstalk risk graph, each node represents a net and any two nodes are connected by an edge if they can be routed next to each other without any crosstalk violation. This violation is checked by making sure that the crosstalk value (sensitivity) between any two nets does not exceed the noise margin of the gates driven by these two nets. Hence an edge $E_{ij}$ in the constrained simple path crosstalk risk graph indicated that the edge weight of $E_{ij}$ does not exceed the noise margins of the gates driven by nets $i$ and $j$. The noise margin of the gates driven by nets $i$ and $j$ (i.e. nodes $i$ and $j$) represents the risk tolerance bounds of nodes $i$ and $j$. The section below briefs a few methods that can be used to obtain the noise margins of the gates in the circuit.
4.1.1.2 Risk Tolerance Bound

The risk tolerance bound is defined as the maximum risk that a wire can be subjected to without affecting its functionality. In our work, we define the risk tolerance bound to be the noise margin of the gates that are driven by the nets. The noise margin values can be obtained from the standard libraries or from the voltage transfer characteristic (VTC) of the gate. The points in the VTC where the slope is \(-1\) can be used to find the \(V_{OH}\), \(V_{OL}\), \(V_{IH}\), \(V_{IL}\) which can then be used to find the noise margins \(NM_H\) and \(NM_L\).

In our work, we show results for noise margin values of 3.5V and 5V. If the sensitivity between two nets is beyond this limit, then those two nets cannot be placed adjacent to each other. If the crosstalk noise seen by a net from both of its adjacent nets is less than the noise margin driven by this net, then the net can be routed adjacent to those two nets.

Risk Tolerance Bound Partitioning:

In [2], there is an additional step which involves partitioning the risk tolerance bound of each net among its routing regions. [2] uses an integer linear programming method to perform bound partitioning. Since our bound criterion is the noise margin of the driven gate, we have avoided the need to perform bound partitioning. By avoiding the need for partitioning the risk tolerance bound, we have been successful in reducing the overall execution time since the ILP method of partitioning would have taken considerable time.
to perform its function. Hence, an expensive time consuming step has been avoided in our approach.

4.1.2 Crosstalk Risk Reduction

Once the crosstalk risk has been estimated using the steps explained in the previous sections, the algorithm proceeds by reducing the risk of each region of the chip as explained in the sections below.

4.1.2.1 Shielding

Shielding is one of the methods used to reduce the crosstalk risk of a region. As explained in the previous chapter, once the CRG\textsubscript{csp} is constructed, the various trees in the forest can be connected using shields. The number of shields needed is equal to the number of disjoint trees in the forest.

Two types of shields can be used for this purpose. The first option is to use the power / ground rails. But since it is not feasible to route the power/ ground rails through all paths of the circuit, this option has its limitations. An alternate way of shielding is to use non-sensitive nets for the purpose. Non-sensitive nets are those nets which are not sensitive to any other net in the region, i.e. if i is a non-sensitive net, for any net j in the region, the element S\textsubscript{ij} is zero.
4.1.2.2 Empty Tracks

If enough shields are not available in a region, an empty track can be used to isolate the nets from each other. But since this option will increase the number of tracks required for routing and hence the routing area, it is avoided whenever possible.

4.1.2.3 Global Routes Re-Adjustment

Net ripping up / re-routing:

If a sensitive net cannot be isolated from the other nets through shielding / empty track, the net is ripped up and re-routed in some other region of the chip. The crosstalk risk of the new region is then estimated again to find out if any violation has occurred. A new route for the net is searched in all regions of the chip. Net ripping up and re-routing is made possible because we have access to the global routing solution and also have the flexibility to work in all regions of the chip.

New global routing solution:

If no such new route can be obtained for a net, the global routing step is performed again to obtain a new global routing solution which is then subjected to all the steps explained in this chapter. The process is repeated till an acceptable solution is obtained. Since global routing is performed through simulated annealing based Timberwolf 3.2 package,
a new solution is generated on every run providing a new global routing solution to work on every time.

### 4.2 Detailed Routing

The post global routing crosstalk optimization algorithm explained above produces a crosstalk free net order for every routing region. “Compared to the original global routing solution before optimization, this risk-free solution is a much better starting point for a crosstalk-driven detailed router to generate a risk-free routing solution of the chip since many problems at the detailed routing level due to crosstalk violations have been eliminated at an earlier stage”[2]. If there are no vertical constraints in the routes of the nets in the chip, the net order can directly be used to obtain the final routing solution. But this solution will be a trivial and in-efficient one since the number of tracks in the region will be equal to the number of nets in the region.

To make better use of the various advantages of channel routing like track reduction and dog-legging, a crosstalk aware detailed routing phase has to be developed which will make use of the crosstalk free net order in conjunction with the channel routing techniques to obtain an efficient final routing solution.

To summarize, the post global routing optimization process provides a crosstalk free net order for each routing region of the chip which will ensure that the routing solution of the chip is crosstalk free. The algorithm presented in [2] has been further improved to make it
more optimistic in its approach. The various improvements proposed in our work have been presented in the next section.

### 4.3 An Improved Approach

We made the following improvements to the algorithm presented in [2]

1) In our approach, we use noise margin as the objective function. Since noise margin directly impacts the circuit functionality, our approach produces more realistic results that can be used for all circuits.

2) Sensitivities are not assumed to be user defined as in [2]. Instead, they are generated by using the model explained in the previous sections.

3) In our work, the sensitivities between each pair of nets are unique. They are not assumed to be ‘1’ or ‘0’. Instead, each net pair has a differing sensitivity (any real number) calculated in the model developed.

4) In the construction of the CRG (crosstalk risk graph), a net is included in the sensitivity list if it is sensitive to at least one other net in the chip. This has been improved to include only the nets that are sensitive to any other net in that routing region (channel). Hence, the number of nodes in the CRG is considerably reduced.
5) The bound is distributed to each region in the original implementation. This involved a computation expensive step namely “risk tolerance bound partitioning”. In our approach, the need for the bound partitioning is avoided and hence the execution time is improved.
Chapter 5

Results and Conclusion

The algorithms explained in Chapter 4 and Appendix A were implemented using C++ in SUN SOLARIS UNIX machines. GCC compiler was used for compiling the C++ files. Shell scripting was used for handling the large amounts of source code files. GDB was used for debugging the code. The final outputs were written to text files and also converted to .mag format in order to view the layout in the MAGIC layout editor.

The algorithms were tested using 10 different standard cell benchmarks having a range of 10 to 2000 cells with 15 to 2500 nets respectively. The results for the various benchmarks have been presented in the sections that follow.
5.1 Results

As explained in Chapter 1, the major objectives of this research work are the following:

- To arrive at a crosstalk free routing solution to ensure failure-free operation of the circuit.
- To improve upon existing approaches to crosstalk synthesis in order to achieve accurate crosstalk estimation and reduction.
- To achieve considerable improvement in performance with a small increase in the overhead.
- To speed up the existing approaches to crosstalk optimization in order to minimize the execution time overhead.

The results presented in this chapter will demonstrate the experiments conducted to achieve the objective functions mentioned above. Ten benchmark suites with different cell and net configurations were used for conducting the experiments. Three responses were measured for each benchmark circuit.

**All the benchmarks were subjected to the following algorithmic steps**

1) Placement using simulated annealing based Timberwolf3.2 package.
2) Global routing using simulated annealing based Timberwolf3.2 package.
3) Post global routing crosstalk optimization using the method developed in [2].
4) Post global routing crosstalk optimization using the improvements suggested in our research work.

5) Detailed routing for the crosstalk un-optimized global routing solution using channel routing algorithm. (Method 1)

6) Detailed routing for the crosstalk optimized post global routing solution generated by using the algorithm in [2] using channel routing algorithm. (Method 2)

7) Detailed routing by channel routing for the crosstalk optimized post global routing solution generated by using the improvements proposed in this work. (Method 3)

The results obtained from steps 5, 6 and 7 are then compared with each other to validate our claims. Each table will thus have 3 columns indicating the results obtained in each of these methods. It will be shown in the subsequent sections that the algorithms presented in [2] and the improvements suggested in this thesis will produce superior results compared to a crosstalk un-optimized routing solution obtained by performing the algorithm in step 5.

The capacitance and resistance values were obtained from IBM 90nm process parameters. The supply voltage (VDD) for this technology is 1V. The results obtained for the various benchmarks have been presented below. The results were verified for two different noise margin (NM) values. Here, we assume that the noise margin high (NM_H) is equal to the noise margin low (NM_L). For an ideal inverter, NM_H = NM_L = VDD/2 where VDD is the supply voltage. Hence NM_H = NM_L = 0.5V for an ideal inverter.
The results obtained for NM = 0.5V has been presented in the sections below. The standard guaranteed value of noise immunity for CMOS circuits is 30-45% of supply voltage [17]. Typical noise margin value for a 90nm process varies from 38% VDD to 44% VDD. Hence, we also present results obtained for the noise margin value of 0.35V.

The following responses were measured:

<table>
<thead>
<tr>
<th>CROSSTALK REDUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Total crosstalk Value (expressed in voltage change sum)</td>
</tr>
<tr>
<td>• Average crosstalk value per net segment (expressed in volts)</td>
</tr>
<tr>
<td>• Maximum glitch value (expressed in volts)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AREA OVERHEAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Number of routed tracks</td>
</tr>
<tr>
<td>• Area (expressed in square lambda (where lambda = 0.045 microns))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPEED UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Execution times (expressed in HH:MM:SS format)</td>
</tr>
</tbody>
</table>

These results will be presented for the following three methods

• Crosstalk un-optimized routing
• Crosstalk optimization using [8]
• Crosstalk optimization using the methods proposed in this research work
5.1.1 Crosstalk Reduction

We express the crosstalk reduction obtained by performing the optimization steps using three metrics

- Total crosstalk value
- Average crosstalk value
- Maximum glitch value

5.1.1.1 Total Crosstalk Value (TCV)

The algorithm implemented in [2] and the improvements suggested in this thesis help in arriving at a net order for routing the nets in the chip such that they are crosstalk-risk free. In other words, the crosstalk that exists after the post global routing optimization is less than the risk tolerance bound of the nets. It should be noted that there still exists an amount of crosstalk between the nets that are routed using the net order even though they are below the bound and hence will not cause any crosstalk risk violation.

In order to compare the algorithm used in [2] and the improvements made in this work, we estimate the total amount of crosstalk that is prevalent in the chip after performing the crosstalk optimization. We call this measure as the ‘total crosstalk value’. This is measured by summing up the sensitivities (crosstalk risk) of each net segment of the chip in their final route. Since each sensitivity value is nothing but the change in voltage, we define a unit called ‘voltage change sum’ to denote the total change in voltage value.
Tables A1 and A2 show the total crosstalk value obtained for all three cases namely, routing without crosstalk optimization, routing with crosstalk optimization using algorithm in [2] and routing with crosstalk optimization using methods proposed in this thesis. The results are shown for noise margin values of 0.5V and 0.35V as explained previously.

As can be observed from the tables, the total crosstalk value obtained from the crosstalk optimization using our method (marked in bold in the tables) are smaller than the values obtained by routing without crosstalk optimization and the routing with crosstalk optimization using the algorithm in [2]. This can be better observed from the graphs shown in Figures 5.1 and 5.2.

![Figure 5.1: Graph showing the % TCV reduction (for NM = 0.5V) for various circuit sizes compared to the TCV values obtained from a crosstalk un-optimized approach.](image-url)
It can be noted from the curves the total crosstalk value reduction is almost the same for both the noise margin values (0.5V and 0.35V) using our approach.

![Graph showing the % TCV reduction (for NM = 0.35V) for various circuit sizes compared to the TCV values obtained from a crosstalk un-optimized approach.](image)

But in the case of the algorithm employed in [2], the reduction obtained for NM = 0.35V is higher than the values obtained for NM = 0.5V. This is because in our work, the

---

**Table 5.1: Line equation and R-squared value for the plots in Figures 5.1 & 5.2**

<table>
<thead>
<tr>
<th>Noise Margin</th>
<th>T.Xue et al</th>
<th>Our Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM = 0.5V</td>
<td>$y = 2.3728x + 4.4255$</td>
<td>$y = 3.6049x + 35.91$</td>
</tr>
<tr>
<td></td>
<td>$R^2 = 0.4556$</td>
<td>$R^2 = 0.4839$</td>
</tr>
<tr>
<td>NM = 0.35V</td>
<td>$y = -0.5183x + 28.288$</td>
<td>$y = 2.6206x + 43.922$</td>
</tr>
<tr>
<td></td>
<td>$R^2 = 0.0179$</td>
<td>$R^2 = 0.3643$</td>
</tr>
</tbody>
</table>
algorithm tends to produce the best possible routing solution in terms of crosstalk reduction irrespective of the upper bound of the noise value. In [2], the algorithm tries to satisfy the bound value by accepting any one solution that meets the criteria and hence settling for a local optimum rather than finding the global optimum. If the NM value was set to a value lower than 0.35V, both the algorithms will proceed in such a way so as to obtain routing solutions which will satisfy the new bound. As showed later in this chapter, this additional constraint will result in an increase in the area.

The line equation and the R-squared values for the two plots have been reported in Table 5.1. It can be seen that the slope of the curves obtained for our method is higher than the slope obtained for the plots of [2]. Hence as the circuit sizes increase, the results obtained from the approach employed in this research produces superior results than the ones obtained using the approach presented in [2].

5.1.1.2 Average Crosstalk Value (ACV)

The average crosstalk value is a measure of the average crosstalk per net segment. It is expressed in volts. This value indicates the average noise amplitude of the various segments of all the sensitive nets in the chip. The ACV is obtained by taking the average of the total crosstalk value (TCV) over all the net segments in the chip.

Thus, \[ \text{ACV} = \frac{\text{TCV}}{\text{Total # of sensitive net segments}} \] (5.1)
Tables A3 and A4 present the ACV values obtained for the various benchmarks for all three methods using NM = 0.5V and NM=0.35 V respectively.

![Graph showing the % ACV reduction (for NM = 0.5V) for various circuit sizes compared to the ACV values obtained from a crosstalk un-optimized approach.](image)

**Figure 5.3**: Graph showing the % ACV reduction (for NM = 0.5V) for various circuit sizes compared to the ACV values obtained from a crosstalk un-optimized approach.

<table>
<thead>
<tr>
<th>Noise Margin</th>
<th>T.Xue et al</th>
<th>Our Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM = 0.5V</td>
<td>( y = 4.9112x - 11.946 ) ( R^2 = 0.679 )</td>
<td>( y = 6.6985x + 19.85 ) ( R^2 = 0.8924 )</td>
</tr>
</tbody>
</table>

**Table 5.2**: Line equation and R-squared value for the plot in Figure 5.3

From the readings, it can be observed that the results obtained using the improvements suggested in this research work achieves a considerable reduction in the average crosstalk value. It can also be seen that the average values in column 5 (with crosstalk optimization by using the algorithm in [2]) is higher than the ones obtained in column 4 (with no crosstalk optimization). This is due to the fact that the algorithm in [2] concentrates...
primarily on meeting the crosstalk risk bounds and in that process settles for local optimum solutions which satisfy the criteria, whereas the method proposed in this work always tries to achieve a global optimum as explained in the previous section. It can also be observed that the average crosstalk value tends to show a gradual decrease as the circuit size increases owing to the fact that the number of sensitive net segments increases with the circuit size.

Table 5.2 lists the line equation and R-squared value for the plot in Figure 5.3. It can be observed from the slopes of the two curves that the ACV reduction obtained by our approach and the approach in [2] increases as the circuit size increases. The % increase is more for our approach than the one obtained for [2].

5.1.1.3 Maximum Glitch Value (MGV)

The maximum glitch value (MGV) is a measure of the worst case amplitude of crosstalk glitch observed in the circuit. Tables 5.3 and 5.4 show the MGV values for all the ten benchmarks.

From Table 5.3, we can infer that the peak amplitude of noise obtained in our work is lower than the ones obtained from the crosstalk-unaware routing and the optimization obtained in [2]. This notion can be further confirmed by observing the maximum glitch for NM = 0.35 shown in Table 5.4. The glitch amplitude settles at a value close to 0.35 in
column 5 because of the fact that the algorithm in [2] accepts any solution which meets the bound criteria.

<table>
<thead>
<tr>
<th>Benchmark No:</th>
<th># of Cells</th>
<th># of Nets</th>
<th>Without Crosstalk Optimization (volts)</th>
<th>With Crosstalk Optimization (volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Using T.Xue et al Method</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>0.286143</td>
<td>0.334173</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>0.235731</td>
<td>0.333685</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>0.320404</td>
<td>0.342348</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>45</td>
<td>0.386143</td>
<td>0.342699</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>120</td>
<td>0.390544</td>
<td>0.345050</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>150</td>
<td>0.400280</td>
<td>0.345387</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>750</td>
<td>0.447225</td>
<td>0.345685</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1250</td>
<td>0.464117</td>
<td>0.359460</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>1000</td>
<td>0.462930</td>
<td>0.346576</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2500</td>
<td>0.472818</td>
<td>0.378901</td>
</tr>
</tbody>
</table>

Table 5.3: Maximum glitch value (in volts) for noise margin = 0.5V and VDD = 1V

<table>
<thead>
<tr>
<th>Benchmark No:</th>
<th># of Cells</th>
<th># of Nets</th>
<th>Without Crosstalk Optimization (volts)</th>
<th>With Crosstalk Optimization (volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Using T.Xue et al Method</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>0.286143</td>
<td>0.234072</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>0.235731</td>
<td>0.333685</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>0.320404</td>
<td>0.342348</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>45</td>
<td>0.386143</td>
<td>0.342699</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>120</td>
<td>0.390544</td>
<td>0.345054</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>150</td>
<td>0.400280</td>
<td>0.345145</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>750</td>
<td>0.447225</td>
<td>0.345685</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1250</td>
<td>0.464117</td>
<td>0.339364</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>1000</td>
<td>0.462930</td>
<td>0.343574</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2500</td>
<td>0.472818</td>
<td>0.348003</td>
</tr>
</tbody>
</table>

Table 5.4: Maximum glitch value (in volts) for noise margin = 0.35V and VDD = 1V
It can be seen from column 4 (crosstalk un-optimized circuit) that the values for benchmark 5 to benchmark 10 (marked in italics) violate the noise margin constraint of the gates and hence will lead to spurious logic transitions. It is also evident from the above table that the crosstalk optimization methods in column 5 and column 6 make sure that the maximum glitch amplitude is not more than the noise bound (0.35V) thereby ensuring correct operation of the circuit. The results obtained so far show that the routing solution obtained using the method in [2] and using the improvements suggested in this research work ensure the correct operation of the circuit by maintaining the crosstalk noise values to the allowable limit. This is achieved at the expense of increase in area and execution time. These parameters are analyzed in the sections that follow.

5.1.2 Area Overhead

5.1.2.1 Number of Routed Tracks

The crosstalk optimization algorithms explained in Chapters 3 and 4 distribute the nets among its routes such that the crosstalk risk is minimized and is below a bound. In doing so, there might be empty tracks formed between nets in order to avoid the crosstalk between those adjacent nets. Also, the nets will take alternative routes that might result in an increase in the number of tracks in a channel. The increase in the number of tracks will in turn increase the area. So track count is a good measure to compare the area efficiency of the algorithms.
Table 5.5 lists the track counts obtained for all the 3 methods for noise margin value of 0.5V.

<table>
<thead>
<tr>
<th>Benchmark No:</th>
<th>Without Crosstalk Optimization (# of tracks)</th>
<th>With Crosstalk Optimization (# of tracks)</th>
<th>With Crosstalk Optimization (% Increase in # of tracks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22</td>
<td>35</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>68</td>
<td>91</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>78</td>
<td>106</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>387</td>
<td>459</td>
<td>418</td>
</tr>
<tr>
<td>6</td>
<td>431</td>
<td>482</td>
<td>448</td>
</tr>
<tr>
<td>7</td>
<td>3599</td>
<td>3697</td>
<td>3603</td>
</tr>
<tr>
<td>8</td>
<td>10155</td>
<td>10966</td>
<td>10180</td>
</tr>
<tr>
<td>9</td>
<td>7650</td>
<td>7699</td>
<td>7671</td>
</tr>
<tr>
<td>10</td>
<td>26681</td>
<td>27403</td>
<td>27104</td>
</tr>
</tbody>
</table>

Table 5.5: Track count for noise margin = 0.5V and VDD = 1V

Table 5.5 confirms the notion that the track count without crosstalk optimization will always be lower than or equal to the ones obtained after optimization for crosstalk. This is a tradeoff that needs to be considered while optimizing for improving the performance. But it can also be seen from columns 5 and 6 that the track count increase from the un-optimized value is smaller for optimization using the methods suggested in our research than the ones obtained by using the method in [2]. This can be seen much clearly in the column showing the % increase in the number of tracks.

It is worthwhile to note that the increase in the number of tracks by performing crosstalk optimization is marginal and the performance improvement obtained by the crosstalk synthesis offsets this marginal increase thereby providing an overall gain.
5.1.2.2 Area

The increase in track count is a direct indication of increase in routing area. Since a small increase in silicon area can cause a significant rise in the manufacturing cost of a chip, careful consideration should be given when trading off the area for performance. The output from all the benchmarks are written to an output file and read using the magic layout editor to obtain the layout. The area of this layout was then measured and reported in Table A5. The results show that there is an increase in area for the circuits with crosstalk optimization when compared to the crosstalk un-aware circuit, but this increase is negligible as shown in the graph in Figure 5.4 below. It should also be noted that the increase in area reported for the optimization using algorithm in [2] is more compared to the area increase with improvements suggested in this research work.

![% Increase in Area](image)

Figure 5.4: Graph showing the % increase in area for the two crosstalk optimization approaches compared to an un-optimized circuit
It should be noted that for circuit sizes of 45 and 1000, the increase in area for the algorithm developed in [2] is close to 5%. The results for our work tend to produce a maximum area increase of 3%. The area measurements for NM = 0.35V is not reported here because of the fact that the values don’t deviate much from the ones shown above.

5.1.3 Speed Up

5.1.3.1 Execution Times

The algorithm developed in [2] used a different objective function to bind the crosstalk values which necessitated a risk-tolerance bound partitioning step which is very expensive in terms of computation time. We avoid the need for this step in our approach thereby achieving a speed up of operation. The results in Table 5.6 show the execution times required by the two approaches for performing the post global routing crosstalk optimization process only.

<table>
<thead>
<tr>
<th>Benchmark No:</th>
<th># of Cells</th>
<th># of Nets</th>
<th>With Crosstalk Optimization (HH:MM:SS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using T.Xue et al Method</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>00:00:05</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>45</td>
<td>00:00:06</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>120</td>
<td>00:00:24</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>150</td>
<td>00:00:26</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>750</td>
<td>00:01:12</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1250</td>
<td>00:03:51</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>1000</td>
<td>00:02:57</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2500</td>
<td>00:07:43</td>
</tr>
</tbody>
</table>

Table 5.6: Execution times for the two crosstalk optimization algorithms
For large circuit sizes, the speed up using our approach is considerable. For example, for benchmark 10, our method takes 44% less time than the one implemented in [2].

5.2 Summary of Results and Conclusion

This research aimed at improving an existing crosstalk optimization algorithm which produces a failure free routing solution for the chip. The focus was mainly on developing a new model for representing the crosstalk information of the nets and achieving more accurate results by using a new objective function. The results presented earlier in this Chapter demonstrated that the model helped in achieving better results than the previous approach. Significant speed-up was also observed. The major observations that were made during the course of this research are

1) Crosstalk optimization by using the method proposed in [2] and the improvements suggested in this work result in a crosstalk free net order which will ensure the proper operation of the circuit.

2) The crosstalk reduction (measured using total crosstalk value, average crosstalk value and the maximum glitch) obtained using the algorithm described in this research work is better than the values obtained for the crosstalk optimization procedure in [2].

- The total crosstalk value obtained by our method was always lesser than the value obtained for [2] with a maximum reduction of 69% and an average reduction of 44%. 
• The average crosstalk value showed a similar trend as above with a maximum reduction of 70% and an average reduction of 59%.

• The maximum noise amplitude obtained by our approach was 15% of VDD with an average of 14% of VDD. For the approach in [2], the values were 37% and 33% of VDD respectively. Hence, it can be seen that our approach produces better noise reduction than the approach in [2].

3) The crosstalk optimization comes at the expense of increased area compared to the crosstalk un-aware approach. But it can be seen that the increase in area for optimization by our method produces a negligible increase in area.

• Our approach produces a maximum of 3% increase in area whereas the approach in [2] gives a maximum of 5%.

4) The total number of tracks (and in turn the area) obtained by the method described in this work is fewer than the ones obtained for the optimization using [2].

• Our approach produces a maximum of 22% increase in area with an average of 7% compared to [2] which gives a maximum of 37% with an average of 16%.

5) Our approach achieves considerable speed-up from [2] owing to the new objective function we use which avoids the time critical steps used in their approach.

• We get a speed up of 44% for the largest sized benchmark compared to the work in [2].
Chapter 6

Future Work

In this thesis, we improved upon an existing capacitive crosstalk synthesis algorithm that works in a post global routing stage to perform crosstalk optimization and reduction. As IC feature sizes continue to reduce to the deep submicron dimensions, additional factors that affect the performance and integrity of the circuit come into the picture. The research reported in this thesis can be modified and extended to include these related effects to provide accurate results. A few possible enhancements to our work are mentioned in this chapter.
1) As the clock frequency of ICs increase, crosstalk due to inductive coupling starts to affect the performance of the chip. Hence, in addition to capacitive crosstalk, inductive effects need to be accounted for these circuits.

2) Effects due to process variations, supply voltage reduction and statistical parameters impact the integrity of the chip and will become prominent as the device dimensions are further reduced. These factors should be considered to make the estimation and reduction of noise more accurate.

3) As the clock frequency increases, the resistance of wire will no longer be linear and constant. Due to skin effect, the resistance becomes frequency dependent. Hence accurate measurement of resistance requires the knowledge of the skin effect when operating in high frequencies.

4) In this work, we performed optimization for standard cell circuits. This can be extended for use with macro cells and IO pads to achieve more versatility.

5) To measure the change in voltage (sensitivity) value, we developed a formula which uses a lumped RC model. This can be further enhanced for accuracy by making use of more accurate distributed RC and RLC network models.

6) Crosstalk affects both the functionality and delay of a circuit. In our work, we analyzed the effect of crosstalk on the logic levels of the circuit called ‘glitch analysis’.
The effect of crosstalk violations on timing and delay can be accommodated by observing the transitions of the signals at various points of time during the circuit operation. This can be achieved by using a timing window to monitor the signal transitions. The impact of crosstalk noise on timing can be estimated and reduced by observing the timing window.
Bibliography


Mudd College and Rutgers University, Spring 2005.

Placement and Global Routing Package”, Proceedings of the 1986 Design Automation

http://vlsicad.ucsd.edu/courses/ece260b-w04/crosstalk-tutorial.pdf

[21] Xiong J J,Chen J,Ma J,He L., “Post global routing RLC crosstalk budgeting” Proc

[22] Lei He , Kevin M. Lepak, Simultaneous shield insertion and net ordering for
capacitive and inductive coupling minimization, Proceedings of the 2000 international
symposium on Physical design, p.55-60, May 2000
Appendix A

Initial Placement and Routing

The initial placement and global routing will serve as the input to the algorithm developed in our research. The placement / global routing algorithm will not change the end result of this work and hence any existing algorithm can be used for performing the same. Standard Cells were used throughout the experiments. Standard cell layout style is such that the cells are arranged in horizontal rows with pads placed around the periphery of the chip. Placement of standard cells consists of allocating the cell in rows on the chip. The TimberWolf3.2 standard cell placement and global routing package was used for performing the initial placement and crosstalk un-aware global routing.
A1. Timberwolf3.2: Standard Cell Placement and Global Routing

In the TimberWolf3.2 Package, the Placement and Global Routing proceeds over 3 distinct stages [19].

(i) In the first stage, TimberWolf3.2 uses the general combinatorial algorithm known as Simulated Annealing to place the cells such that the total estimated interconnect cost is minimized.

(ii) During the second stage, Timberwolf3.2 inserts feed through cells as necessary and the minimization of the total estimated interconnect cost proceeds again in the manner of simulated annealing. The second stage comes to a close following a global routing step, in which the number of wiring tracks needed is accurately estimated.

(iii) During the third and final stage, local changes are made to the placement whenever such changes result in a reduction in the number of wiring tracks required.

A1.1 Stage 1: Standard Cell Placement

The Initial Cell Placement was done using Simulated Annealing based TimberWolf3.2 Package. The pseudo code of the algorithm is as shown in Figure A1.
The acceptance of a new state $j$ is determined by $\text{accept}$, whose structure is shown below.

The acceptance function is governed by the function $f$ given by

$$f(\Delta c, T) = \min \{ 1, e^{-\Delta c/T} \} \quad (A1)$$

The control parameter above is the temperature $T$ whose update function is given by

$$T_{\text{new}} = \alpha T_{\text{old}} \quad 0 < \alpha < 1 \quad (A2)$$
The simulated annealing algorithm used above is characterized by (1) The generate function \textit{generate}, (2) The acceptance function \textit{accept}, (3) The updating function \textit{update} (4) The \textit{inner loop criterion} and (5) The \textit{stopping criterion}.

The actual implementation of these functions in terms of the standard cell placement is explained below:

- **The Generate Function:**

  The function generate is responsible for selecting the next new state and it does so by making a weighted random selection from one of the following actions [19]. (1) The displacement of a single cell to a new location, possibly on a different row. (2) The interchange of two cells or (3) An orientation change for a cell. The selection of alternative 3 occurs only for 1/10\textsuperscript{th} of the number of times the alternative 1 resulted in a new state being rejected.

  The ratio \( r \) of single cell displacements to pairwise interchanges “\( r \)” is kept at 4. The generation function in TimberWolf3.2 is controlled by a range limiter window. The actual formulae controlling the respective window dimensions are shown in Equations A3 and A4.

  \[
  \frac{\text{windowX}}{2} = \frac{1}{6} \times \text{xspan} \times \log_{10}(\varepsilon T) \tag{A3}
  \]
  \[
  \frac{\text{windowY}}{2} = \frac{1}{6} \times \text{yspan} \times \log_{10}(\varepsilon T) \tag{A4}
  \]

  The value of \( \varepsilon \) was kept at 0.2.
• **The Cost Function:**

The cost function for the simulated annealing algorithm of stage 1 consists of 2 independent functions C1 and C2 [19].

\[
C_1 = \sum_{\text{net}=1}^{\text{numnets}} [\text{xspan}] + [\text{yspan}] \quad \text{(A5)}
\]

\[
C_2 = \sum_{r=1}^{\text{numRows}} |\text{actRowLen}[r] - \text{desRowLen}[r]| \times \text{parameter} \quad \text{(A6)}
\]

The value of the parameter in C2 is set at 5 to create a minimum penalty of 5. Cell overlaps were avoided to simplify the algorithm. Linear overlap penalty function was not used in this implementation since no overlaps were allowed.

• **The Inner Loop Criterion**

The inner loop criterion in TimberWolf3.2, that is, the number of iterations of the inner loop, is specified in terms of the number of attempted new states per cell per value of T (att.per.cell). The value of att.per.cell was varied from 10 to 200 for the different benchmarks tested. The number of iterations of the inner loop for a given value of T is then given by: numiter = att.per.cell \times N, where N is the number of standard cells.

• **The Control of Temperature T:**

The function \textbf{update} (T) is expressed by

\[
T_{\text{new}} = \alpha(T_{\text{old}}) \times T_{\text{old}} \quad 0 < \alpha < 1 \quad \text{(A7)}
\]
The value of $\alpha$ for various values of $T$ is given below:

<table>
<thead>
<tr>
<th>$T$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>40000</td>
<td>0.80</td>
</tr>
<tr>
<td>20000</td>
<td>0.84</td>
</tr>
<tr>
<td>10000</td>
<td>0.88</td>
</tr>
<tr>
<td>5000</td>
<td>0.91</td>
</tr>
<tr>
<td>200</td>
<td>0.94</td>
</tr>
<tr>
<td>100</td>
<td>0.85</td>
</tr>
<tr>
<td>50</td>
<td>0.85</td>
</tr>
<tr>
<td>5</td>
<td>0.80</td>
</tr>
<tr>
<td>1.5</td>
<td>0.70</td>
</tr>
<tr>
<td>0</td>
<td>0.10</td>
</tr>
</tbody>
</table>

Table A1: $\alpha$ value for different temperatures

- **The Stopping Criterion:**

  Timberwolf3.2 automatically terminates its simulated annealing algorithm as $T$ becomes less than 0.1.

**A1.2. Stage 2: Placement Refinement**

As $T$ is reduced below $T_0$, the TimberWolf3.2 program enters the first part of stage 2, which continues the simulated annealing portion of the placement methodology. Before continuing with the simulated annealing, the feed through path requirements are ascertained and feed-through cells are inserted to create the required path.

“The generation of states function (generate) takes on a somewhat different form during stage 2. For every iteration of the inner loop, the following 5 steps are completed: (1) randomly select a cell A amongst all the standard cells and feed through cells. (2) The
next step is to find the left neighbor, L, of the selected cell (A) if such a neighbor exists. (3) Next, find the right neighbor, R, of the selected cell A if such a neighbor exists. (4) If L is null, then the next new state attempted is an interchange of cells A and R. On the other hand, if R is null, then the next new state attempted is an interchange of cells A and L. If neither L nor R is null, one of them is randomly selected to be interchanged with A. If this new state is not accepted, then the next new state is generated by interchanging A with the other neighbor. (5) The next new state attempted is generated by proposing an orientation change for cell A. C1 is the only cost function used in this stage.” [19]

**A1.3. First Stage of the Global Router:**

“The goal of the global router is to select is to select pairs of pins for interconnection such that the total number of wiring tracks required (i.e. the height of the chip) is minimized” [19]. A good approximation of the number of wiring tracks required is the total channel density which is given by:

\[
\text{totChanDens} = \sum_{\text{all Chan's}} \text{density [chan]} \tag{A8}
\]

Where density [chan] is the density of channel chan.

**Definitions** [19]:

**Pin Cluster:** An internally connected (electrically-equivalent) group of pins is referred to as a pin cluster.
**Net Segment**: A portion of a net which must connect two pin clusters is referred to as a net segment.

**Switchable Net Segment**: If each cluster has a pin on the top of the cell as well as a pin on the bottom of the cell, then this net segment is defined as switchable. That is, a switchable net segment can be routed in the channel above the row of the channel below the row to which the row belongs. The algorithm shown in next page is used for performing the global routing function.

At this point, all of the possible edges connecting the nodes have been added to the graph for this particular net. Kruskal’s Algorithm is then applied to generate the minimum spanning tree. For each edge of the tree representing the global route of the specific net, one pin from each cluster is selected to form the actual net segment. In the case of an edge connecting two clusters on the same row, it is determined if this is a switchable net segment. If so, then two pairs for pins are selected, one for routing in the channel above and one for the channel below.

**A1.4. Second Stage of the Global Router:**

The second stage of global router is to select from among the choices presented by the switchable net segments in such a manner as to minimize the total channel density. This stage proceeds using simulated annealing with T being maintained at zero (T = 0).
**Algorithm** Global_Routing

**Begin**

**ForEach** net n **DO**

Identity the pin clusters of net n;

**ForEach** pin cluster P find x(P);

Sort the pin clusters of n on their x-coordinates;

**For**(::)

(*Construct the cluster graph corresponding to net n*)

TorB ← 0

(*TorB = 1, -1, 0 if row(P2) = row(P1) + 1, row(P1) – 1, or row(P1), respectively*)

If clusters = [ ] Then **Exit**;

**EndIf**;

P1 ← Head(Clusters); (*Get the leftmost element if clusters*)

**For**(::)

Find the closest pin cluster P2 to the right of P1 such that row(P2) = row(P1) + 1, row(P1) – 1, or row(P1);

If there is no such P2 Then **Exit**; (*Exit from the innermost FOR loop*)

Else

**Case**

(1) row(P2) = row(P1): TorB = 0;

(*Connect nodes P1 and P2*)

Add edge(P1,P2) to the cluster graph;

Exit;

(2) row(P2) = row(P1) + 1: If TorB != 1 Then

{ Add edge(P1,P2) to the cluster graph;

If TorB = -1 Then **Exit**;

Else TorB = +1;

**End If**;

} **EndIf**;

(3) row(P2) = row(P1) - 1: If TorB != -1 Then

{ Add edge(P1,P2) to the cluster graph;

If TorB = +1 Then **Exit**;

Else TorB = -1;

**End If**;

} **EndIf**;

**End Case**

**EndIf**;

**End For**(::)

End **For**(::)

End **ForEach** net n **DO**

End **Begin**

---

Figure A3: Pseudo-code of the global router [19]
A2. Detailed Routing using a Channel Router

The constrained Left-Edge Algorithm is used for performing the channel routing. The channel to be routed is represented by a rectangular region with two rows of terminals along its top and bottom sides. The channel net list is represented by two vectors TOP and BOT to represent the grid points on the top and bottom sides of the channel. Two-layer HV routing is employed here. In two-layer routing, all horizontal wires are laid out on tracks on one layer and all vertical wires on the other.

A2.1 Constraint Graphs

For any instance of channel routing, two constraint graphs are constructed, one to model the horizontal constraints and one to model the vertical constraints.

**Horizontal Constraint Graph (HCG):** The horizontal constraint graph, denoted by HCG(V,E) is an undirected graph where a vertex \( i \in V \) represents net \( i \) and edge \( (i,j) \in E \) if the horizontal segments of net \( i \) and \( j \) overlap.

**Vertical Constraint Graph (VCG):** The vertical constraint graph in the channel routing problem can be represented by a directed graph VCG(V,E), where each node \( i \in V \) corresponds to net \( i \), and each vertical column introduces an edge \( (i,j) \in E \) if and only if net \( i \) has a pin on the top and net \( j \) on the bottom for the channel in the same column. That is, for any two nets with pins at the same column on opposite sides of the channel, there
will be an edge between their corresponding vertices in the VCG. Therefore, if there is a
cycle in the VCG, the routing requirements cannot be realized without dividing some nets.

A2.2 Constrained Left-Edge Algorithm

The original left-edge algorithm was modified to include the vertical Constraints to get
the constrained left-edge algorithm. In the left-edge algorithm, the segments of nets to be
connected are sorted in the increasing order of their left end points from left-edge of the
channel.

```
Begin
1) Sort all nets on their leftmost end position
2) Select the next net with the lowest left-end position
   If n has no descendents in VCG
      Then Begin
         Place n on the lowermost available track;
         Delete n from the sorted list;
         Delete n from VCG
      End
   Else Goto 2
   EndIf
3) Continues scanning the sorted list and from it select those nets which do not overlap
   with the nets assigned to this track and have no descendents in VCG; Remove all
   selected nets from the list
4) If list != Empty Then Goto 2
5) Exit
```

Figure A4: Pseudo-code of the left-edge algorithm based channel router
Appendix B

Tabulation of Results

The tabulations for the following results are presented in this appendix:

- Total crosstalk value (TCV) for NM = 0.5V and VDD = 1V
- Total crosstalk value (TCV) for NM = 0.35V and VDD = 1V
- Average crosstalk value (TCV) for NM = 0.5V and VDD = 1V
- Average crosstalk value (TCV) for NM = 0.35V and VDD = 1V
- Area of the circuit for NM = 0.5V and VDD = 1V
B1. Total Crosstalk Value (TCV)

<table>
<thead>
<tr>
<th>Benchmark No:</th>
<th># of Cells</th>
<th># of Nets</th>
<th>TCV Without Crosstalk Optimization (in VCS)</th>
<th>TCV With Crosstalk Optimization (in VCS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using T.Xue et al Method</td>
<td>Using OUR Method</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>3.35704</td>
<td>2.82852</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>2.40361</td>
<td>2.20638</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>12.4979</td>
<td>11.7477</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>45</td>
<td>14.5840</td>
<td>12.5258</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>120</td>
<td>101.943</td>
<td>89.4388</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>150</td>
<td>107.472</td>
<td>85.7420</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>750</td>
<td>1127.85</td>
<td>1038.58</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1250</td>
<td>3806.52</td>
<td>2277.35</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>1000</td>
<td>2802.81</td>
<td>1989.43</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2500</td>
<td>10546.5</td>
<td>8321.08</td>
</tr>
</tbody>
</table>

Table B1: TCV (in VCS) for noise margin = 0.5V (Ideal logic) and VDD = 1V

<table>
<thead>
<tr>
<th>Benchmark No:</th>
<th># of Cells</th>
<th># of Nets</th>
<th>TCV without Crosstalk Optimization (in VCS)</th>
<th>TCV with Crosstalk Optimization (in VCS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using T.Xue et al Method</td>
<td>Using OUR Method</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>3.35704</td>
<td>1.95793</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>2.40361</td>
<td>1.97168</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>12.4979</td>
<td>9.27817</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>45</td>
<td>14.5840</td>
<td>9.37289</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>120</td>
<td>101.943</td>
<td>81.3071</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>150</td>
<td>107.472</td>
<td>95.821</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>750</td>
<td>1127.85</td>
<td>1021.41</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1250</td>
<td>3806.52</td>
<td>2198.90</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>1000</td>
<td>2802.81</td>
<td>1979.91</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2500</td>
<td>10546.5</td>
<td>8319.80</td>
</tr>
</tbody>
</table>

Table B2: TCV (in VCS) for noise margin = 0.35V (Ideal logic) and VDD = 1V
### B2. Average Crosstalk Value (TCV)

<table>
<thead>
<tr>
<th>Benchmark No:</th>
<th># of Cells</th>
<th># of Nets</th>
<th>ACV Without Crosstalk Optimization (in volts)</th>
<th>ACV With Crosstalk Optimization (in volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using T.Xue et al Method</td>
<td>Using OUR Method</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>0.176687</td>
<td>0.314280</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>0.171686</td>
<td>0.320638</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>0.198380</td>
<td>0.192505</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>45</td>
<td>0.199780</td>
<td>0.192516</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>120</td>
<td>0.270405</td>
<td>0.276046</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>150</td>
<td>0.355277</td>
<td>0.276089</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>750</td>
<td>0.315305</td>
<td>0.291736</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1250</td>
<td>0.375989</td>
<td>0.214526</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>1000</td>
<td>0.367871</td>
<td>0.209450</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2500</td>
<td>0.395935</td>
<td>0.275689</td>
</tr>
</tbody>
</table>

Table B3: ACV (in volts) for noise margin = 0.5V (Ideal logic) and VDD = 1V

<table>
<thead>
<tr>
<th>Benchmark No:</th>
<th># of Cells</th>
<th># of Nets</th>
<th>ACV Without Crosstalk Optimization (in volts)</th>
<th>ACV With Crosstalk Optimization (in volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using T.Xue et al Method</td>
<td>Using OUR Method</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>0.176687</td>
<td>0.226322</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>0.171686</td>
<td>0.328614</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>0.198380</td>
<td>0.299296</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>45</td>
<td>0.199780</td>
<td>0.312430</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>120</td>
<td>0.270405</td>
<td>0.286293</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>150</td>
<td>0.355277</td>
<td>0.288340</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>750</td>
<td>0.315305</td>
<td>0.298483</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1250</td>
<td>0.375989</td>
<td>0.215243</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>1000</td>
<td>0.367871</td>
<td>0.210454</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2500</td>
<td>0.395935</td>
<td>0.279092</td>
</tr>
</tbody>
</table>

Table B4: ACV (in volts) for noise margin = 0.35V (Ideal logic) and VDD = 1V
## B3. Area of the circuit (in Sq. Lambda)

<table>
<thead>
<tr>
<th>Benchmark No</th>
<th># of Cells</th>
<th># of Nets</th>
<th>Without Crosstalk Optimization (Sq. Lambda)</th>
<th>With Crosstalk Optimization (Sq. Lambda)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using T.Xue et al Method</td>
<td>Using OUR Method</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>271350</td>
<td>272250</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>202020</td>
<td>202020</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>646800</td>
<td>674505</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>45</td>
<td>704375</td>
<td>742450</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>120</td>
<td>1593900</td>
<td>1624805</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>150</td>
<td>1934400</td>
<td>2010100</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>750</td>
<td>23854400</td>
<td>24754000</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1250</td>
<td>41695500</td>
<td>42195060</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>1000</td>
<td>34200000</td>
<td>35921065</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2500</td>
<td>113864375</td>
<td>114135400</td>
</tr>
</tbody>
</table>

Table B5: Area of the circuit (in Sq. Lambda) for noise margin = 0.5V