2014

Computation of cross-talk alignment by mixed integer linear programming

Qifeng Chen

Department:

COMPUTATION OF CROSS-TALK NOISE ALIGNMENT

BY

MIXED INTEGER LINEAR PROGRAMMING

by

QIFENG CHEN

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER ENGINEERING

2014

Approved by

Yiyu Shi, Advisor
Minsu Choi
Jun Fan

ABSTRACT

Noise analysis has been an important and difficult part of design flow of very large-scale integrated (VLSI) systems in many years. In this thesis, the problem of signal alignment resulting in possible maximum peak interconnect coupling noise and propose a variation aware technique for computing combined noise pulse taking into account timing constraints on signal transitions has been discussed. This work shows that the worst noise alignment algorithm can be formulated as mixed integer programming (MLIP) problem both in deterministic window cases and variational window cases. For deterministic window cases, it is assumed that timing windows are given for each aggressor inputs and the victim net is quite. It compares the results from proposed method with the most known and widely used method for computing the worst aggressor alignment – sweeping line algorithm, to verify its correctness and efficiency. For variation window cases, as variations of process and environmental parameters result in variation of start and end points of timing windows, linear approximation is used for approximating effect of process and environmental variations. One of the biggest advantages of MILP formulation of aggressor alignment problem has also been discussed, which is the ability to be easily extended to more complex cases such as non-triangle noise pulses, victim sensitivity window and discontinuous timing windows, this work shows that such extension can be solved by algorithm and does not require development of new algorithms. Therefore, this novel technique can handle noise alignment problem both in deterministic and variational cases and can be easily extended for more complex cases.

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my advisor Dr. Shi, who helped me a lot during my graduate study, both in academic and in personal life, thank you a lot.

I would also like to thank my committee members: Dr. Choi and Dr. Fan, from whom I learned a lot.

Thanks to my lab mates, the atmosphere in lab is a boost to my graduate study, thank you, Dian Ma, Geng Hui, Tao Wang, Li-Chia Chen, Jianming Liu, Weizhi Meng, Uma, Khalid, Jinglan Liu, Ying Zhang.

Finally, thank my girlfriend, Chunyu Wang, who has made my life interesting and meaningful. Last but not the least; thank my family for always being supportive.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Page

LIST OF TABLES

Page

1. INTRODUCTION

This section is organized as followed; it will cover the background of this problem in Subsection 1.1, the problem formulation in Subsection 1.2, and the literature review in Subsection 1.3.

1.1. BACKGROUND

Continuing scaling of critical dimensions, reduction of supply voltage and increase of complexity and density digital circuits makes signal integrity one of the major problems in very large-scale integrated (VLSI) design.

Noise analysis has been an important and difficult part of chip design flow for many years [1]. In this study, the problem of how crosstalk noise is affected by the switching times of aggressors acting on a victim net has been discussed. It is assumed that the driver strengths, wire spacing, spatial positions of aggressors, and the victim are given and not changing. Signal arrival times can be adjusted to achieve the maximum peak noise.

Increasing variability of transistor and interconnect characteristics makes noise analysis today a even more challenging problem. Process and environmental variations affect cross-talk noise in two ways. First, transistor and interconnect characteristics and their variations can decide the size and the shape of the noise pulses. Second, the variations change the time of signal transitions, which affects their relative alignment and consequently the size of the total noise pulse.

The variability of signal delays accumulates along signal propagation paths. The variability of signal switching times can be significantly larger than signal slews. Therefore, process and environment variations have a huge impact on the signal alignment thus affects the size of noise pulse. Moreover, it can move noise pulse out of victim sensitivity interval, which means reducing the noise impact to zero. Therefore, ignoring variability in noise analysis results in overestimation of crosstalk effects of VLSI chips.

1.2. PROBLEM FORMULATION

Crosstalk noise occurs when a wire with switching signal affects another wire through coupling capacitance between these wires. The net that is affected is called victim net, on the other hand, the nets that injecting coupling noise into victim net are called aggressor nets. A circuit that consists of a victim net and several aggressor nets and their coupling capacitances is called a noise cluster, shown in Figure 1.1.

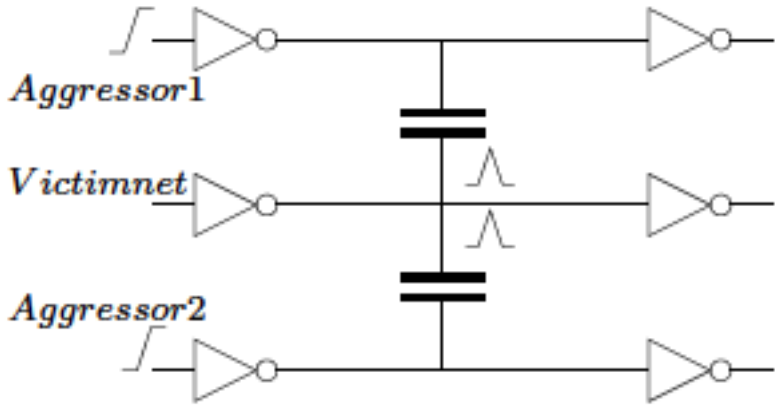

Figure 1.1 Circuit of Noise Cluster

Usually in noise cluster circuit, a victim net is coupled to more than one aggressor nets. The total noise injected into the victim net is a combination of the pulses injected by all individual aggressors. And usually the aggressors do not switch simultaneously. Therefore, this noise analysis tool must compute the worst possible noise pulse that is possibly created by all the aggressor nets. To compute its worst possible noise pulse requires aggressor net's switching times. However, the exact moment of the signal transitions in a circuit are never known because it depends on the so many factors such as gate and interconnect delays and their variations, crosstalk noise of the previous stage and so on. Therefore, each net can only switch inside certain timing interval, called switching window, which is calculated by timing analysis in the form of the earliest and latest

signal arrival times. So in order to computes worst possible alignment of aggressor, it is assumed that each net can only switch inside its timing windows.

1.3. LITERATURE REVIEW

The problem of the worst aggressor alignment always attracted attention of EDA research community. A unified method of computing the worst noise alignment is to formulate as mixed nonlinear integer programming problems [2]. Unfortunately, there is no known efficient algorithm of solving general nonlinear mixed integer programming problem.

Sweeping Line Algorithm is the most known and widely used method for computing the worst aggressor alignment in deterministic cases [4]. It has *O(nlgn)* complexity so it is very effective and efficient. In Section 2.1, it will implement the Sweeping Line Algorithm as the comparison of the proposed method.

However, both the mixed nonlinear integer programming algorithm and the sweep line algorithm deal with process and environment variations by expanding timing windows. The solution can be over pessimistic because it might create false overlap of timing windows, which can cause overestimation of cross-talk noise.

There also are methods dealing with timing windows overlap in statistical timing analysis [8]. However, the proposed approach computes only probability distribution of bounding window overlap, and does not solve the problem of worst alignment noise. It is basically Monte-Carlo technique and can be rather inefficient for large designs.

2. NOISE ALIGNMENT FOR DETERMINISTIC WINDOW

This section is organized as followed, it will talk about the sweep line algorithm, which will act as the comparison of this algorithm in Subsection 2.1, and it will discuss the detail of this MILP algorithm in deterministic cases in Subsection 2.2. Subsection 2.1 will cover the modeling, worst coupling noise and the superposition, the noise alignment with timing constraints, the algorithm, and the complexity of the sweep line algorithm. Subsection 2.2 will cover the background, modeling and the detail MILP formulation.

2.1. SWEEP LINE ALGORITHM

2.1.1. Modeling. In sweep line algorithm, the problem formulation is that for the convenience of computing the worst alignment noise, make every individual peak noise occurring at the same time, and it is achievable by changing the aggressor arrival time. For example, assuming there are five aggressors represented by *A1, A2, A3, A4, A5,* respectively, in Figure 2.1. *P1, P2, P3, P4, P5* are the peak amplitude of the noise generated by each aggressor at the victim's output, respectively. Figure 2.1. shows the typical timing relationship between aggressor arrival times and peak noise occurring times. As it is shown, when make all aggressors have the same arrival time, peak noise occurring time usually cannot have the same arrival time.

But also timing constraints need to be taken into account, which is, every aggressor has a timing window which is bounded by its early mode arrival time and late mode arrival time by timing analysis, so the problem became to find a location of an imaginary sweep line, as shown in Figure 2.2. For the given timing windows such that the total contribution of crosstalk noise from each intersected aggressor is maximum at this location.

When no timing constraints considered, the aggressor alignment determines the relative arrival times. On the other hand, when timing constraints considered, the aggressor alignment determines the absolute values of the arrival times. The first aggressor alignment is the foundation of the second aggressor alignment because the shape of the sweep line is obtained through the first alignment.

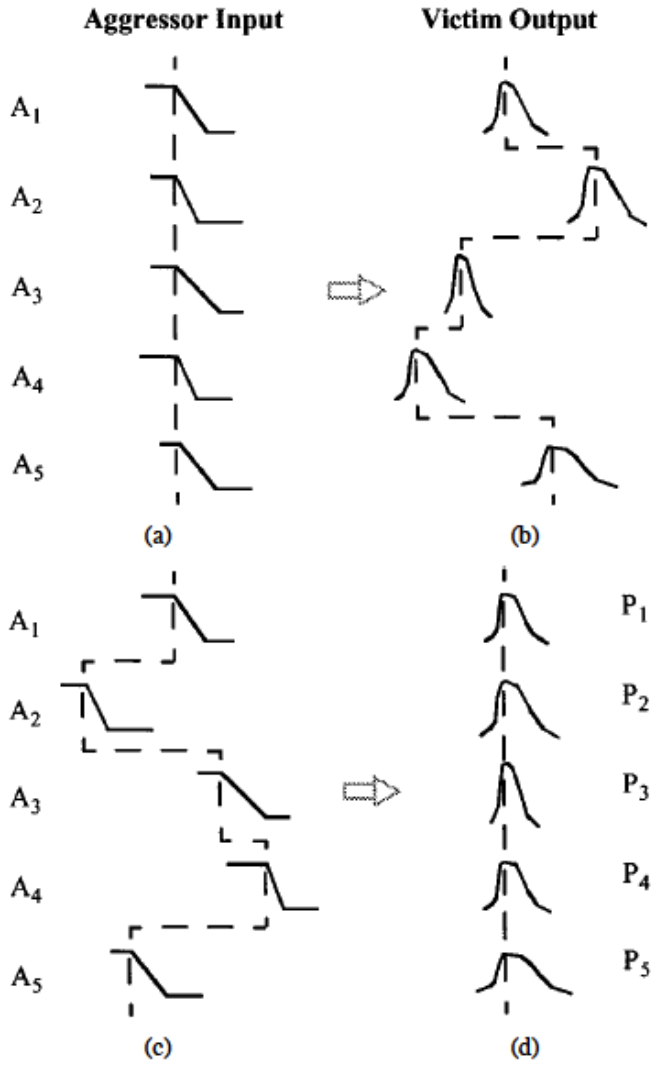

Figure 2.1 Alignment of Aggressor Inputs Versus Alignment of Victim Peak Noise [4]

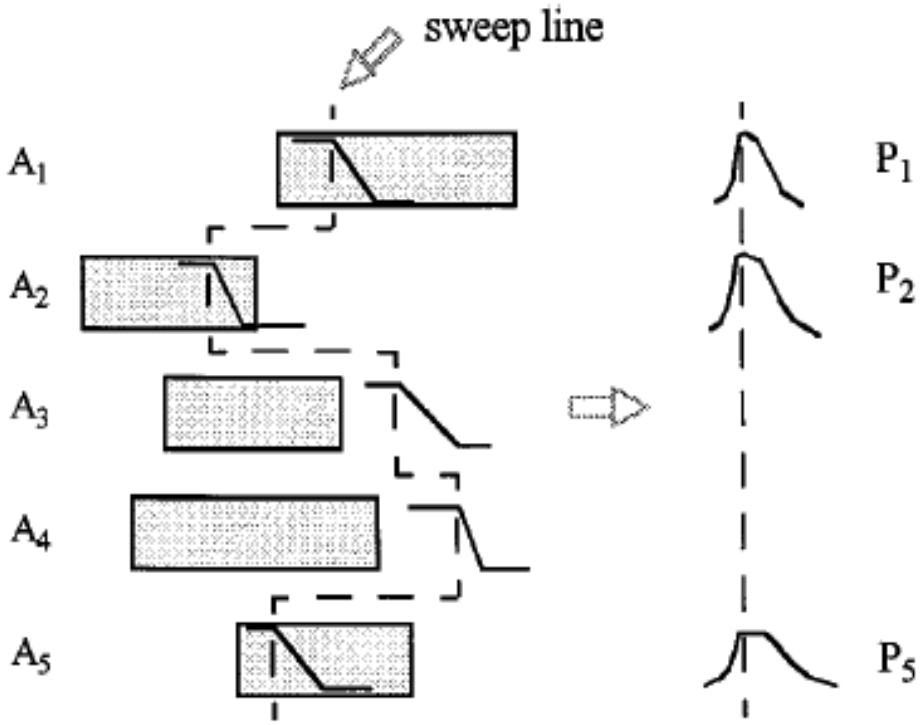

Figure 2.2 Timing Window Versus Aggressor Alignment [4]

2.1.2. Worst Coupling Noise and Superposition. Simulating designs with complicated nonlinear driver models is often too much time and resource consuming. Linear driver models are used in fast noise estimations.

Superposition holds for linear models [4]. When the noise amplitude becomes larger, the coupling network is no longer purely linear and that will cause inaccuracy when use superposition principle to do noise estimation. A modified linear network with a piecewise linear victim driver model can capture the nonlinear voltage dependency.

In this section, it will discuss the relationship between the actual worst coupling noise and the peak noise obtained from applying the superposition. For a relatively large crosstalk, it will show the trend compared with smaller noise.

Four possible WCN analysis strategies based on aggressor alignment method is listed. Assume *m* is the number of aggressors; *M* is the total number of noise calculation (excluding addition).

A. Explicit aggressor alignment (exhaustive search). Noise output waveform is obtained by properly aligning switching of all aggressors with appropriate skew between inputs. The largest amplitude of this type is WCN. Usually $M>>m$.
B. No aggressor alignment (zero skew). Noise output waveform is obtained for simultaneous switching of all aggressors with zero skew between their inputs. This is a special type of explicit aggressor alignment with zero skew. $M=1$.
C. Implicit aggressor alignment (superposition). Noise output waveform is obtained by applying superposition principle. Each individual noise is obtained when only one aggressor is injecting while all others are quiet, and aligned such that their peak amplitudes occur at the same time. Total peak noise is the summation over every individual peak noise. $M= m$.
D. Extension of the implicit aggressor alignment (sweep line algorithm). Compared to method three, instead of adding the individual noise waveforms, it "back-annotates" the signal skews, implied by the alignment of individual peak noise to each aggressor's inputs. And use those skewed aggressor inputs to simulate or estimate the coupling stage again. $M=m+1$.

Figure 2.3 shows that about 31% difference in peak amplitude between Method A and Method B and about 23% difference in peak amplitude between Method A and Method C. Method A always has the largest peak noise. And sweep line algorithm is usually a close approximation to Method A.

The result tells us that the direct addition of individual peak noise can cause significant inaccuracy in worst coupling noise estimation. However, there is no clear boundary as to when the overestimation will occur or when the underestimation will occur. To avoid overestimation or underestimation induced by simple application of superposition, two further steps are necessary: explicit aggressor alignment to find the proper arrival times and recalculation to obtain the total peak noise. Due to the accuracy tolerance in signal skew estimation, a simple method is used in sweep line algorithm to

find the required arrival times: the first step is back-annotation of individual peak noise occurring time to the corresponding aggressor arrival time; the second step is recalculation of total crosstalk noise when every aggressor is switching, with the new arrival time. To increase accuracy, iteration can be applied until no more changes in arrival times occur.

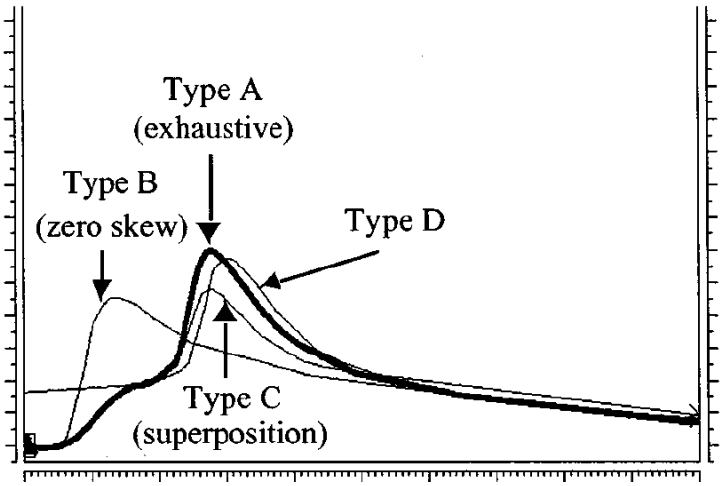

Figure 2.3 Comparison of Four Aggressor Alignment Methods [4]

2.1.3. Aggressor Alignment with Timing Constraints. In this section, the worst-case coupling analysis for a quiet victim when timing constraints are specified is considered. And to be more accurate, a more accurate formulation is needed.

First, introduce the concept and the formula for effective pulse width (EPW) [4] of a noise waveform. EPW is a measure of the range of a noise waveform. Given a noise waveform $v_0(t)$, its EPW is defined as follows:

$$EPW = \frac{\int_0^{\infty} t v_0 \, dt}{\int_0^{\infty} v_0 \, dt} \tag{1}$$

There exists an easy yet efficient formula to estimate EPW.

$$EPW = \sum_{C_i \in C} C_i R_{ii} \quad (2)$$

C is the set of interconnect capacitance and loading capacitance for he coupling circuit, including line-to-ground capacitance and coupling capacitance. C_i is the *i*th capacitance in *C*. R_{ii} is the equivalent resistance (including driver resistance and interconnect resistance) seen across capacitor C_i when all other capacitances are open. In other words, EPW can be estimated by the sum of open circuit time constants.

By simulation, it is found that the width of the noise pulse cannot be neglected and the actual shape of noise pulse is not always "sharp" at its top. Therefore, partial contribution (when the peak noise is not perfectly aligned) of each crosstalk noise which has been ignored.

So the aggressor alignment can be reformulated as explained in Figure 2.4. Figure 2.4(a) shows the original timing window and sweep line. In Figure 2.4(b), the sweep line has been straightened. Therefore, the timing windows have been moved to satisfy the following condition: a line sweep in (a) is equivalent to that of (b), in terms of vertical intersections with particular aggressor windows. In Figure 2.4(c), timing window has been expanded to include the width of the noise pulse. The total expanded portion for each timing window is the corresponding pulse width EPW.

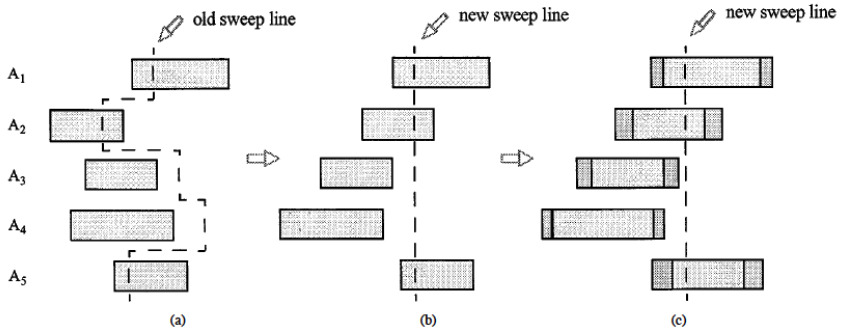

Figure 2.4 Reformulation of Aggressor Alignment [4]
(a) Original Timing Window. (b) Stretched Timing Window.
(b) (c) Expanded Timing Window.

So the problem becomes:

Each timing window shown in Figure 2.4(c) is considered as a line segment n the channel. And the width of the timing window corresponds to the length of the segment. The peak noise is a weight of the original portion of the segment. The expanded portion has a weight equal to a fraction of the peak noise. The leftmost and rightmost points of the expanded window have zero weights. A linear function is used to approximate the weight in the expanded portion.

2.1.4. Algorithm. Step 1: Compute the peak amplitude (P_i), its occurring time (t_i), and the EPW for each individual waveform. $W_L^{(i)}$ and $W_R^{(i)}$ are defined as

$$W_L^{(i)} = \rho \cdot EPW_i \tag{3}$$

$$W_R^{(i)} = (1-\rho) \cdot EPW_i \tag{4}$$

Where r is an experimental data and $0 < r < 0.5$.

Step 2:

Adjust timing windows, as shown in Figure 2.5.

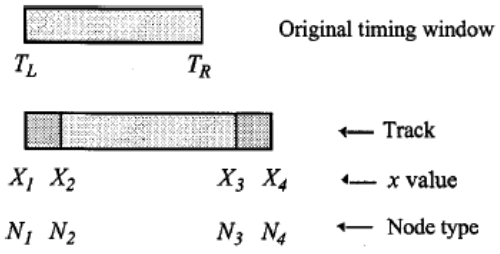

Figure 2.5 Timing Window Adjustment (Stretched and Expanded)

From this step, it obtained 4n nodes in total (n aggressors). Each node is associated with a x-coordinate value, a segment number and a node type. The fur node types have been defined in Figure 2.5. where N_2 and N_3 denotes left and right endpoints of the stretched timing window, respectively, the N_1 and N_4 denote the left and right endpoints of the expanded timing window. The *i*th segment represents the timing window for the *i*th aggressor. The x-coordinate value is defined by

$$X_1^{(i)} = T_L^{(i)} + T_i - W_L^{(i)} \tag{5}$$

$$X_2^{(i)} = T_L^{(i)} + t_i \tag{6}$$

$$X_3^{(i)} = T_R^{(i)} + t_i \tag{7}$$

$$X_4^{(i)} = T_R^{(i)} + t_i + W_R^{(i)} \tag{8}$$

Step 3: Weighted Channel Density (WTCD) Algorithm.

begin

sort the nodes lexicographically on x-coordinates such that Node[1] is leftmost and Node[4n] is rightmost.

Max = 0; (maximum density)

Cur = 0; (current density)

Xmax = 0; (x-coordinate value for Max)

LTree = 0; (a balanced tree to store the point of the left expanded window)

RTree = 0; (a balanced tree to store the point of the right expanded window)

x_1 = 0; (current x-value)

for k=1 *to* 4n *do*:

(Let *i* and X be he segment number and x-coordinate value associated with *N*)

N = Node[k];

$x_2 = x_1$; (previous x value)

$x_1 = X$;

$cur+ = \sum_{j \in LTree} P_j \cdot ((x_1 - x_2) / W_L^{(i)})$ (adjust density)

$cur- = \sum_{j \in RTree} P_j \cdot ((x_1 - x_2) / W_R^{(i)})$ (adjust density)

if $cur > Max$, then $Max = cur$; $X\max = x_1$;

if $N \hat{I} N_1$, then $LTree \rightarrow insert(i)$

else if $N \hat{I} N_2$, then $LTree \rightarrow delete(i)$

else if $N \hat{I} N_3$, then $RTree \rightarrow insert(i)$

else ($N \hat{I} N_4$), then $RTree \rightarrow delete(i)$

end

2.1.5. Complexity. The complexity of the WTCD algorithm is determined by two factors: the first one is the sorting algorithm, the other one is the insert and the delete operations in the for loop. The time complexity of the sorting algorithm is O(nlgn). Within the loop, balanced trees are used to store the endpoints, in these cases, red-black tree is used, so the insert and the delete operation only take O(lgn). Since the for loop can executes at most 4n times, the time complexity of the WTCD algorithm is only O(nlgn).

2.2. MIXED INTEGER LINEAR PROGRAMMING (MILP) ALGORITHM

2.2.1. Background and Modeling. Conventionally noise analysis assumes linear approximation for combined noise pulse injected with several aggressors. According to that model, the combined noise pulse is a linear superposition of the pulses injected by the individual aggressors. As discussed in sweep line algorithm, linear model is sufficiently accurate. Even when higher accuracy is required, the linear model of noise superposition is used for computing initial approximation of the worst aggressor alignment. Without the approximation, the search of the worst alignment using nonlinear model can be too expensive.

As it was mentioned above for each net, timing analysis predicts its switching window where the net can have signal transition. It is assumed that a noise pulse can appear at any time moment inside its switching window.

Therefore, in Mixed Integer Linear Programming Algorithm, the problem of computing the worst aggressor alignment and the corresponding worst noise pulse can be formulated as follows:

Knowing a set of aggressor noise pulses, and a set of aggressor switching windows where the noise pulses may appear compute the highest possible combined

noise pulse and the corresponding alignment of the pulses. In other words, it is required to find such set of the overlapping switching that the linear superposition of the corresponding noise pulses has maximum height. It is possible that the overlap of the timing windows is so small that the corresponding noise pulses are only partially aligned. Most often the injected noise pulses are modeled with triangle or trapezoid. The problem of worst noise alignment is illustrated Figure 2.6, where timing windows are depicted as rectangles on timing axis.

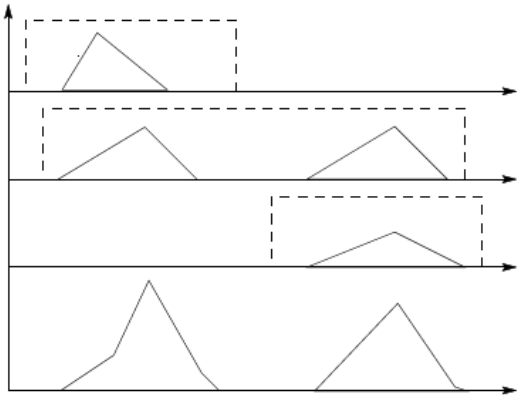

Figure 2.6 Pulse Alignment and Superposition

2.2.2. MILP Formulation in Deterministic Cases. It derived this Mixed Integer Linear Programming Formulation of aggressor alignment problem in Cartesian coordinate system where the abscissa represents time t and the ordinate represents voltage of the noise pulses. First, start this derivation with inequalities expressing the fact that a point G_i with coordinates (t, v_i) lies inside a triangle defined with its tip point $(t_i, h_i > 0)$ and the rise and fall slopes $(r_i > 0)$ and $(f_i > 0)$ of its side, as it is shown in Figure 2.7.

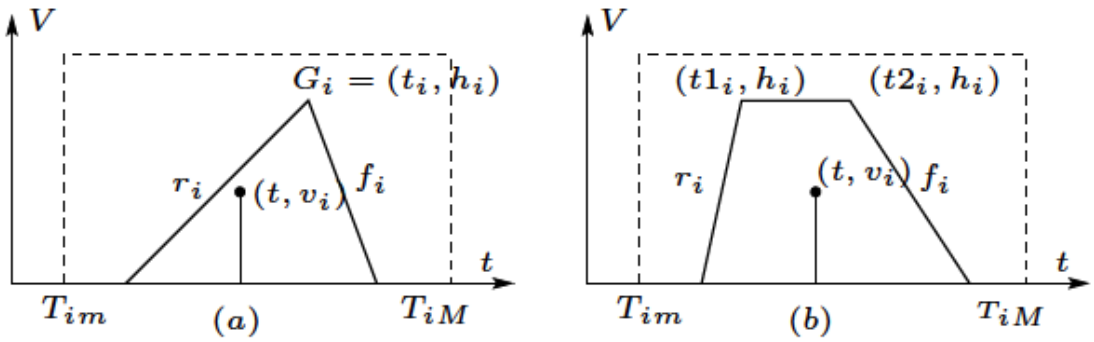

Figure 2.7 Noise Pulse in Switching Window

$$(v_i - h_i) - r_i(t - t_i) \leq 0 \tag{9}$$

$$(v_i - h_i) - f_i(t - t_i) \leq 0 \tag{10}$$

$$v_i \geq 0 \tag{11}$$

Adding switching constraints and assuming N triangular pulses $\Delta_i, i = 1,...,n$, it can write linear program formulation for maximum sum of the ordinates v_i of the points G_i belonging these pulses and laying inside timing windows $W_i = [T_{im}, T_{iM}]$.

$$Maximise_{t_i,t} \sum_{i=1}^{N} v_i$$

Subject to $(v_i - h_i) - r_i(t - t_i) \leq 0$

$(v_i - h_i) - f_i(t - t_i) \leq 0$

$v_i \geq 0$

$T_{im} \leq ti \leq T_{iM}$

The solution of this linear program gives maximum height of the combined noise pulse subject to switching window constraints imposed to each individual noise pulse Δ_i. This linear problem has a solution when there is nonempty intersection of all switching windows expanded by the left and right widths of their noise pulses. Otherwise the switching window constraints are incompatible and the linear program has no solution.

The problem of the worst aggressor alignment requires finding the subset Ω of the intersection active switching windows and their noise pulses contribution to the worst noise pulses. The choice of the subsets can be done through binary variables p_i selecting active windows W_i.

When $p_i = 0$ the switching window W_i is excluded from subset Ω by multiplying the height of the noise pulse Δ_i by $p_i = 0$, reducing the contribution of this pulse into combined noise pulse to 0.The constraints defined by window W_i is relaxed by expanding the window both to left and to right by large value Q. This makes window W_i overlap with all other windows and the constraint is always satisfied. Constant Q is selected to be sufficient larger to provide overlap of any switching windows. For simplicity Q can be set the value to $\max(|T_{iM}|) + \max(w(\Delta_i))$, where $w(\Delta_i)$ is width of pulse Δ_i.

Introducing binary variables p_i into linear program formulation, it is transformed into the following mixed integer linear program (MILP).

$$\underset{t_i, x, p_i}{Maximise} \sum_{i=1}^{N} v_i$$

Subject to $(v_i - p_i h_i) - r_i(t - t_i) \le 0$

$(v_i - p_i h_i) - f_i(t - t_i) \le 0$

$v_i \ge 0$

$T_{im} - Q(1 - p_i) \le t_i \le T_{iM} + Q(1 - p_i)$

This MILP find the amplitude of the largest cumulative noise over all possible combinations of noise pulses satisfying the constraints defined by the switching timing windows.

3. NOISE ALIGNMENT FOR VARIATIONAL WINDOW

Section 3 is organized as followed, Subsection 3.1 will cover the background and the modeling of noise alignment problem for variational window, Subsection 3.2 will cover the MILP formulation for this kind of problem, Subsection 3.3 will cover the branch and cut algorithm which will solve the MILP problem.

3.1. BACKGROUND AND MODELING

Similar to signal arrival times, timing windows depend on process and environmental parameters: Leff, Tox, metal and dielectric thickness, supply voltage, temperature, etc. Variations of beginning and ending times of timing windows are highly correlated with each other. For example, at lower supply voltage, gate delays are higher. Therefore, both start and end moments of timing windows are shifted to the right. Obviously the correlation is not perfect, which complicates the analysis. If noise analysis ignores this correlation, the conservatism of noise estimation can be achieved only by proper expansion of timing windows to encompass their variability. However, the resulted solution will be obviously too pessimistic, which is shown in Figure 3.1 Where is shown that the window expansion created false overlapping of timing windows.

Instead of using Static Timing Analysis, in variational window cases, use Statistical Static Timing Analysis, Statistical Static Timing Analysis computes beginning and ending times of switching windows in a linear canonical forms [15], [16]:

$$t = t_0 + \sum_{i=1}^{n} a_i \Delta X_i + a_R \Delta R \tag{12}$$

Where t_0 is a nominal value of the timing quantity, ΔX_i is a variation of parameter X modeling its chip to chip variability, ΔR is a random variable modeling an uncorrelated variation, a_i and a_R are sensitivities to those variations respectively. This representation contains all information about correlation of switching windows.

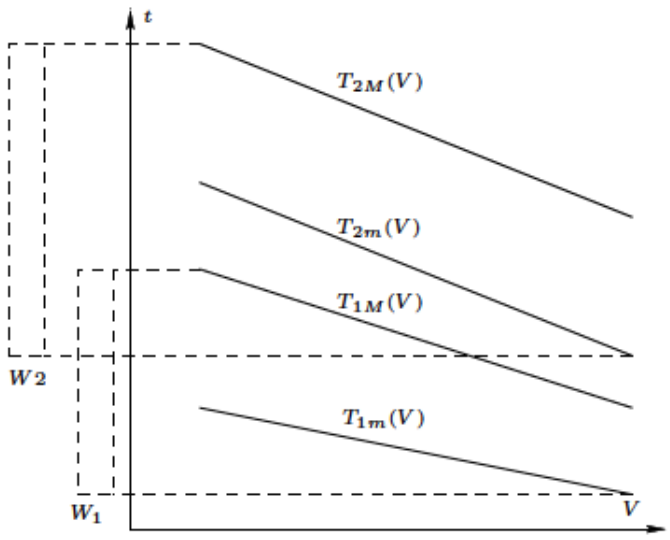

Figure 3.1 False Overlap of Timing Windows

The problem of computing variation aware aggressor alignment can be formulated as follows. Knowing beginning and ending moments of aggressor switching windows expressed in linear canonical forms, variation range of each variational parameter and shape of the noise pulses, compute the worst aggressor alignment and the corresponding worst noise pulse.

Straightforward approach to solving this problem is enumeration of all possible values of variational parameters, computation of worst noise pulse for each of those combinations and selection of the worst pulse among those pulses. Obviously this is a method that is extremely inefficient.

3.2. MILP FORMULATION IN VARIATIONAL CASES

Variations of process and environmental parameters result in variation of start and end points of timing windows. Therefore, T_{im} and T_{iM} are functions of variational parameters X_j. Conventionally, linear approximation is used for approximating effect of

process and environmental variations. Therefore, start and end points T_{im} and T_{iM} of timing windows are expressed within linear forms:

$$T_{im} = T_{im,0} + \sum_{j=1}^{n} a_{im,j} X_j \tag{13}$$

$$T_{iM} = T_{iM,0} + \sum_{j=1}^{n} a_{iM,j} X_j \tag{14}$$

Substituting linear expressions of start and end points T_{im} and T_{iM} into the MILP formulation and adding constraints of variational parameter X_j, it obtain MILP formulation of alignment problem for variational windows.

$$\underset{t_i, x, p_i}{Maximise} \sum_{i=1}^{N} v_i$$

subject to $(v_i - p_i h_i) - r_i(t - t_i) \le 0$

$$(v_i - p_i h_i) - f_i(t - t_i) \le 0$$

$$v_i \ge 0$$

$$T_{im,0} + \sum_{j=1}^{n} a_{im,j} X_j - (1 - p_i) Q \le t_i$$

$$T_{iM,0} + \sum_{j=1}^{n} a_{iM,j} X_j - (1 - p_i) Q \ge t_i$$

$$X_{j,\min} \le X_j \le X_{j,\max}$$

Solving this problem it is found that the maximum noise peak satisfying constraints of switching windows and constraints on process variations, the solution gives also the time moments when the maximum noise occurs a set of binary variables having value 1, which defines the set of overlapping timing windows with maximum noise, and values of variational parameters resulting in maximum noise. Analyzing the LP problem that is a part of the MILP solution a set of variation parameters actively restricting noise peak and sensitivities of noise can be got to those parameters. This information is important for finding which variability is critical for noise and how change of the range of variability affects the noise.

3.3. BRANCH AND CUT ALGORITHM

One way to solve MILPs is to enumerate all the integer solutions in the feasible region and individually check each one for optimality. However, as the dimensions of the problem, *n* grows, enumeration becomes NP hard, meaning that the number of feasible integer solutions grow exponentially, instead of growing by order of some polynomial, $p(x) = ax^n + bx^{n-1} + ... + ex^0$, the enumeration will grow by order $p(x) = s^x$ for some function *s*.

As the numbers of aggressors is not very large MILP problem can be solved efficiently. The efficiency can be improved if branch and cut method explores binary variables in the order of the height of the noise pulses. The branch and cut algorithm is a combination of branch and bound algorithm and cutting plane algorithm. Cutting planes algorithm helps quickly get tighter bounds of the solution and reduce the search. For large noise clusters, the branch and bound algorithm can split the set of aggressors into two or more subsets and solving MILP problem for each subset can find an approximate solution.

3.3.1. Branch and Bound Algorithm. To solve using branch and bound algorithm, first need to solve the problem with relaxation, which means solving it by using the Revised Simplex as if there were no integer restrictions. From this it can obtain the relaxation solution z.

Next, pick up a non-integer variable and branch on it. Most commonly branch it on the most fractional variable, which means the one that is closest to half way between its floor and its ceiling.

For example, if the *z* got is $z = (1.25, 3.45, 2.75, 5)$, branch on z_2, because 0.45 is closer to 0.5. By branching a variable, the root problem is taken and two sub-problems or nodes are created. In this problem, in one node add constraint $z_2 \leq 3$, in the other node add the constraints $z_2 \geq 4$. Let *S* be the feasible solution set of the LP relaxation: $S = \{z : Az \leq b, z \geq 0\}$, now have:

$$S_1 = S \cap \{z : z_j \leq \lfloor z_j \rfloor\} \tag{15}$$

$$S_2 = S \cap \{z : z_j \geq \lceil z_j \rceil\} \tag{16}$$

For here each of the branches must be solved and checked for optimality. After solving the original system and branching on the first variable, each created node is eventually either branched or pruned. A node can be pruned in different ways, optimality, bounds, and infeasibility. Continue to solve the problem in the following manner:

Step 1: Infeasibility

Pick a node and solve it with the Dual Simplex Method with updated constraints. If the Dual Simplex is unbounded then the problem is infeasible and the node is pruned by infeasibility. This occurs when the new constraints disagrees with the constraints that are already established.

Step 2: Optimality/Bounds

Check for bounds and optimality.

Step 3: Branching

If z is not in integer form then branch on the variable which is most fractional.

Step 4: Repeat

Pick a new node and repeat starting on Step1, until all nodes are pruned, at which time the node associated with the lower bound is optimal.

3.3.2. Cutting Planes Algorithm. Adding cutting planes to the system is another good way to solve MILPs. Cutting Planes are very fast but unstable. The idea of cutting planes is to cut out of the feasible region based on information from the optimal dictionary. First solve the relaxation in the Revised Simplex Method, take a look at the information from the optimal dictionary and from here it is able to deduce an inequality that will ‘cut’ out of a piece of the feasible region, then add the new cut to the system and re-optimize using the Revised Simplex Method. The process is repeated until the required variables are integers.

4. EXTENSIONS

One of the biggest advantages of MILP formulation of aggressor alignment problem is its ability to be easily extended to more complex cases. Any such extension can be solved by the same optimization package and does not require development of new algorithms.

- NON-TRIANGLE NOISE PULSES

MILP formulation can be constructed for noise pulses of any convex piece-wise linear shape. For example, if noise pulses are trapezoidal as it is shown in Figure 2.7 (b), the top points defined here are $(t1_i, h_i)$ and $(t2_i, h_i)$ with the rise and fall slopes $(r_i \geq 0)$ and $(f_i \leq 0)$ of sides of the trapezoidal, then the formulation can be constructed by following inequalities.

$$(v_i - h_i) - r_i(t1_i - t_i) \leq 0$$

$$(v_i - h_i) - f_i(t2_i - t_i) \leq 0$$

$$v_i \leq h_i$$

$$v_i \geq 0$$

- VICTIM SENSITIVITY WINDOW

As it was mentioned above that a victim net is sensitive to noise only in certain timing interval called victim sensitivity window. These constraints can be included in MILP formulation by adding these:

$$T_{vm,0} + \sum_{j=1}^{n} a_{vm,j} X_j - (1 - p_i)Q \leq t$$

$$T_{vM,0} + \sum_{j=1}^{n} a_{vM,j} X_j - (1 - p_i)Q \geq t$$

- DISCONTINUOUS TIMING WINDOWS

If aggressors belong to different lock domain it is required to consider discontinuous switching windows, such as windows consisting of many timing intervals as it is shown in Figure 4.1.

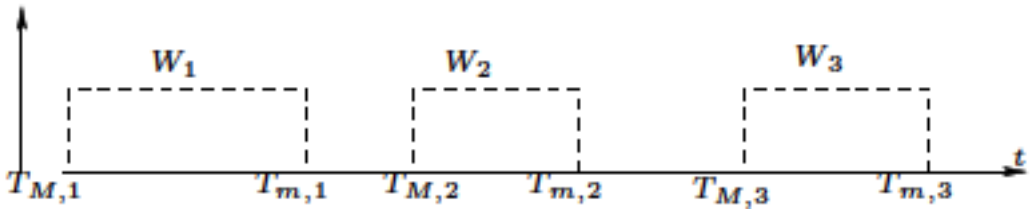

Figure 4.1 Discontinuous Timing Windows

MILP formulation can be modified to consider such cases like followings: IF aggressor net A has discontinuous timing window consisting of K timing intervals $W_1, W_2, ..., W_k$, which can be replaced with K imaginary aggressors $A_1, A_2, ..., A_k$ switching in continuous timing windows and thus can formulate the problem as described and the solution should be the same.

5. EXPERIMENTAL RESULTS

This section shows experimental results of worst coupling noise and its runtime obtained from Mixed Integer Linear Programming (MILP) Algorithm. In deterministic windows, the Sweep Line Algorithm is considered as accurate. For alignment 1, compare the peak noise obtained from proposed method with the results from Sweep Line Algorithm. For alignment 2, compare the results from the method in variational timing window with the results obtained from the max-width timing window, min-width timing window and the nominal deterministic timing window.

Assume the technology is 0.35 um. For Table 5.1, generate eight cases that have one victim and two aggressors for analysis. Each case differs in signal arrival timing window, victim's driving strength, coupling lengths, etc. Table 5.1 shows the comparison between MILP algorithm and sweep line algorithm in deterministic cases. It is shown that the noise voltage of the sweep line algorithm and MILP method are the same.

Table 5.1 Comparison of Sweep Line Algorithm with MILP in Deterministic Cases

	SWEEP LINE		MILP	
Case #	Voltage(V)	Run Time(μs)	Voltage(V)	Run Time(μs)
1	0.2193	144 (1)	0.2193	1166 (8.1×)
2	0.3191	186 (1)	0.3191	866 (4.6×)
3	0.4289	180 (1)	0.4289	736 (4.1×)
4	0.6861	140 (1)	0.6861	625 (4.5×)
5	0.8243	184 (1)	0.8243	1116 (6.1×)
6	1.1867	156 (1)	1.1867	777 (4.9×)
7	1.2567	138 (1)	1.2567	821 (5.9×)
8	1.3674	181 (1)	1.3674	546 (3.0×)

For Table 5.2, five cases that also have one victim and two aggressors are generated, while the timing windows are expressed in linear form, and each case differs from the nominal arrival time, the nominal required arrival time, the sensitivities of multiple variational parameters, Table 5.2 shows the peak noise obtained from the variational timing window and the peak noises obtained from the deterministic cases like the max-width timing window, the min-width timing window and the nominal timing window. Note that the time is in μs. From Table 5.2, it is shown that The MILP algorithm can also deal with variational timing windows, from the experimental results, it is shown that the peak noise generated by the MILP is always greater or equal to the peak noise generated from three different deterministic timing windows, which means the result got from MILP is the largest possible peak noise.

Table 5.2 The Peak Noise Obtained From MILP in Variational Cases Compared with Deterministic Cases on Max-width, Min-width and Nominal Window

	MILP		MAX WIDTH	MIN WIDTH	NOMINAL
Case #	Voltage(V)	Run Time(μs)	Voltage(V)	Voltage(V)	Voltage(V)
1	0.8146	1389	0.8146	0.8146	0.8146
2	0.9582	2367	0.8805	0.9582	0.9117
3	1.1602	2419	1.1545	1.1206	1.1428
4	1.2937	2149	1.2937	1.2937	1.2937
5	1.46374	2261	1.46374	1.3441	1.4107

Note that all the experiments are ran on a machine with following characteristics: 2.5 GHz dual-core Intel Core i5 CPU Turbo boost up to 3.1 GHz, 4GB RAM, the software is as follows: MacOS, GCC.

To sum it up,

- In deterministic cases, MILP algorithm can get the same WCN with sweep line algorithm, which verifies the correctness of MILP on deterministic cases.
- The MILP algorithm can also deal with variational timing windows, from the experiment results, it is shown that the peak noise generated by the MILP is always greater or equal to the peak noise generated from three different deterministic timing windows, which means the result got from MILP is the largest possible peak noise.
- In deterministic cases, the sweep line algorithm works the fastest in existing methods and MILP is not as efficient as it, but MILP can also deal with variational cases and it is easy for more complex situations.

6. CONCLUSIONS

This thesis has developed a novel technique for computing the worst noise alignment for deterministic and variational cases. The method is mainly based on a new way of problem formulation, which leads the problem into a Mixed Integer Linear Programming problem, in which branch and cut algorithm is used to solve efficiently. The correctness of this algorithm is verified with sweep line algorithm, which is now the most efficient method to compute the worst alignment noise in deterministic cases.

This algorithm not only can handle deterministic cases but also are capable of dealing variational cases, and can be easily extended to more complex situations such as non-triangle noises, non-continuous timing windows and so on without requiring any other algorithms.

BIBLIOGRAPHY

[1] R. Levi, D. Blaauw, A. Dasgupta and V. Zolotov, "Clarinet: A noise analysis tool for deep submicron design," in *DAC*, pp. 223-238, 2000.

[2] K. L. Shepard, V. Narayanan, P. C. Elmendorf and G. Zheng. "Global Harmony: Coupled noise analysis for full-chip RC interconnect networks," in *DAC*, pp. 139-146, 1997.

[3] R. Kumar. "Interconnect and noise immunity design for the Pentium 4 processors," in *DAC*, pp. 938-943, 2003.

[4] L. H. Chen and M. Marek-Sadowska. "Aggressor alignment for worst-case crosstalk noise," in *IEEE Trans. on CAD*, pp. 612-621, 2003.

[5] R. Gandikota, D. Blaauw, and D. Sylvester. "Modeling crosstalk in statistical static timing analysis," in *DAC*, pp. 974-979, 2008.

[6] A. B. Kahng, B. Liu, X. Xu. "Statistical timing analysis in the presence of signal integrity effects," in *IEEE Trans* on CAD, pp. 1873-1877, 2007.

[7] S. Shrivastava, and H. Parameswaran. "Improved timing windows overlap check using statistical timing analysis," in *International Conference on VLSI Design*, pp. 70-75, 2011.

[8] H. Fatami, and P. Tehrani. "Crosstalk timing window overlap in statistical timing analysis," in *ISQED*, pp. 245-251, 2013.

[9] D. S. Chen, R. Batson and Y. Dang. "Applied Integer Programming: Modeling and Solution," Joe Wiley and Sons, 2011.

[10] S. Sirichotiakul, V. Zolotov, R. Levy and D. Blaauw. "Driver modeling and alignment for worst-case delay noise," in *DAC*, pp. 720-725, 2001.

[11] C. Visweswariah, K. Ravindran, K. Kalafala, S. Narayan and S. G. Walker. "First-order incremental block-based statistical timing analysis," in *DAC*, pp. 331-336, 2004.

VITA

My name is Qifeng Chen, I was born in Xiantao, Hubei, China. I earned my bachelor degree in Electrical Engineering at Beijing University of Chemical Technology at 2012. Currently, I plan to obtain my Master Degree in Computer Engineering from Missouri University of Science and Technology in Dec 2014.