December 2014

Discrete Gate Sizing Methodologies for Delay, Area and Power Optimization

Jiani Xie
Syracuse University

Abstract

The modeling of an individual gate and the optimization of circuit performance has long been a critical issue in the VLSI industry. In this work, we first study of the gate sizing problem for today's industrial designs, and explore the contributions and limitations of all the existing approaches, which mainly suffer from producing only continuous solutions, using outdated timing models or experiencing performance inefficiency.

In this dissertation, we present our new discrete gate sizing technique which optimizes different aspects of circuit performance, including delay, area and power consumption. And our method is fast and efficient as it applies the local search instead of global exhaustive search during gate size selection process, which greatly reduces the search space and improves the computation complexity. In addition to that, it is also flexible with different timing models, and it is able to deal with the constraints of input/output slew and output load capacitance, under which very few previous research works were reported.

We then propose a new timing model, which is derived from the classic Elmore delay model, but takes the features of modern timing models from standard cell library. With our new timing model, we are able to formulate the combinatorial discrete sizing problem as a simplified mathematical expression and apply it to existing Lagrangian relaxation method, which is shown to converge to optimal solution. We demonstrate that the classic Elmore delay model based gate sizing approaches can still be valid. Therefore, our work might provide a new look into the numerous Elmore delay model based research works in various areas (such as placement, routing, layout, buffer insertion, timing analysis, etc.).

Discrete Gate Sizing Methodologies for Delay, Area and Power Optimization

by

Jiani Xie

B.S., Fudan University, 2008
M.S., Syracuse University, 2011

Dissertation
Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering.

Syracuse University
December 2014

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to those who have given me support through my doctoral study.

Foremost, I would like to thank my advisor Professor C.Y. Roger Chen. To me, he is a modern day Socrates. I am so blessed to have his guidance and patience over the years. It is his vast intelligence and immense professional perception that has brightened and motivated me in my research, and it is his enthusiastic personality and charisma that has inspired and will always inspire me in my future career and life. I am forever indebted to him, and the dissertation would not be possible without him.

I would also like to thank all my dissertation committee members for reviewing my work and giving me valuable suggestions. They are (in alphabetic order) Prof. Pinyuen Chen, Prof. Brandon Choi, Prof. Ehat Ercanli, Prof. Duane Marcy and Prof. Qinru Qiu.

And I am blessed to have collaborated with my peers who accompanied me through my study. I am very thankful to Chaobo Li for his insightful discussion for my work in Chapter 5.

My last and everlasting gratitude goes to my dearest parents, who give me endless love and hope in my bad times. Thank everything that happened in my life, whatever good or bad, makes who I am now.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Gate sizing technique is an important technique for improving circuit performance. Researchers have been working on the gate sizing problems for more than thirty years. They have built different timing models, and developed numerous kinds of algorithms to solve this problem.

In this chapter, we will have a detailed review of the gate sizing problem, including the models that researchers have applied for circuit performance evaluation, and the techniques that have been developed so far.

1.1 Gate Sizing Problem

Gate sizing problem is to assign each gate with a proper cell from a standard cell library in a way that it will meet certain design requirement or optimize the circuit performance, such as circuit delay, layout area or power consumption, etc. Gate sizing is a flexible and powerful method to reduce circuit delay, minimize layout area, and optimize statistical power consumption, etc. Besides, it is also versatile in circuit design flow as it may be used to correct timing errors, fix design rule violations and setup and hold violations, fix noise constraints due to crosstalk, improve yield, etc [1].

As circuit performance can be evaluated from different aspects, the gate sizing problem may vary with different optimization objectives. When the timing closure is the first priority, we may want to solve the constraint free gate sizing problem for delay

minimization. And we may solve the power minimization problem with delay constraint if it is the first priority to reduce the power consumption. Hence, there are extensive research works focusing on different optimization objectives with different gate models.

1.2 Gate Model

The way we model the gate and evaluate the delay is of vital importance in gate sizing problems. Over simplified gate model may be easier to solve and implement, but it may not be accurate with the real circuit. However, if the model is too complicated, it may be too hard to solve the problem and require extra runtimes even though it has high accuracy. So we need to balance between the accuracy and performance in modeling the circuit. In this part, we will have a short review of the popular models that previous researchers have used.

First of all, the most common way is to model a circuit as a resistance-capacitor circuit (RC). In a RC circuit, all the gates and interconnects are modeled as components with resistors and capacitors only, then the analysis of the circuit is the same as the analysis of the resistor and capacitor behaviors.

Among the studies of measuring the delay of a RC circuit [2, 3, 5], Elmore delay [2] is widely used because of its simplicity. Elmore delay approximates the circuit delay by the first order impulse response. And the delay of each gate is to multiply the gate resistor with its downstream capacitance, which is easy to implement. And it is proved to be an absolute upper bound on the actual 50% delay on an RC tree response [4]. Then for better accuracy, AWE [5] are usually applied for higher order moments.

Although Elmore delay in RC circuit has been efficient in solving sizing problems, especially for continuous gate sizing problems, it has usually been criticized for being over simplified [38, 39, 40] and lacking waveform information. And as current VLSI industry has more design requirements with more constraints, such as transition time or slews, it is hard to incorporate Elmore delay to propagate slews, and thus we need more accurate models to solve the sizing problem.

Table 1. 1. Inverter Cell in01m01 Information

Cell Name	**in01m01**
Leakage Power	4*uW*
Area	1
Max Output Capacitance	14.4*ff*
Pin capacitance	1*ff*

Table 1. 2. Fall Delay Lookup Table for Inverter Cell in01m01.

Output load capacitance are in *ff*, and *input transition time* are in *ps.*

Output Load Capacitance	Input Transition Time							
	5	30	50	80	140	200	300	500
0	10.17	16.33	20.22	24.81	32.40	38.73	47.43	61.24
1	14.64	20.85	25.59	31.17	39.94	47.43	58.09	75.00
2	19.10	25.31	30.29	36.81	46.76	55.00	67.08	86.60
4	28.03	34.24	39.21	46.66	58.91	68.65	82.49	106.07
8	45.89	52.10	57.07	64.53	79.35	92.03	109.30	137.49
16	81.60	87.81	92.79	100.24	115.16	130.07	153.54	190.32
32	153.03	159.24	164.21	171.67	186.59	201.50	226.36	275.35

Nowadays, the gates are characterized by complex timing models from standard cell libraries. And the gate delay is influenced by more factors compared with earlier models. So it is almost impossible to represent the gate delay with a simple equation like Elmore delay. And each gate may come with a few options of cells, as cells of the same type have the same functionality but different timing and power consumption characteristics. Then instead of using a formula, modern cell libraries are using lookup tables to represent the cell delay. For example, in Table 1. 1 and Table 1. 2, we shows an example of inverter cell in01m01 from ISPD 2012 standard cell library [6]. The basic cell information is listed in Table 1. 1 and the fall delay lookup table is shown in Table 1. 2. The fall delay is a function of input transition time and output load capacitance. Standard cell libraries also have lookup tables for rise delay, fall delay, rise transition time or fall transition time, which provides more accuracy for each cell in library. And it is more flexible in propagating transition time, and cover more cell features. Obviously, since it is not easy to represent the gate model in formula, it creates more mathematical challenges for us to solve the gate sizing problems.

1.3 Previous Work

Researchers have been studying the gate sizing problems for a long time, and many of the earlier works are based on continuous cell sizes [7 - 22].

The continuous gate sizing problem traced back to [7], where Ruehli et al. proposed a method for logic gate delay assignment to achieve power minimization. Then Fishburn and Dunlop [8] presented a sensitivity-based greedy heuristic, which iteratively adjusts the transistors sizes to minimize area or circuit delay. Sapatnekar et al. [9] used convex

programming to solve the sizing program, which is guaranteed to find the exact solution. In [10], Chen et al. used Lagrangian relaxation to solve constrained area minimization problem, which allowed future researchers to apply Lagrangian relaxation to solve sizing problems with different optimization objectives [11, 12, 13]. Kasamsetty et al. [14] developed an accurate convex delay model and applied convex programming technique with the new delay model. Later, Boyd et al. [15] formulated the problem as geometric program and solved it with geometric programming.

For the continuous gate sizing problems, all the techniques can be separated into several categories. Linear programming is used by [16, 17, 18] to work with linear models. And for posynomial models, convex programming is used by [9, 14, 19, 20]. Menezes et al. presented a quadratic programming approach in [21]. And a network flow based algorithm is proposed by Ren and Dutt in [22].

These continuous gate sizing methods assume that gate sizes can be any continuous value, which is unrealistic in real life. So they all require rounding the continuous solutions to discrete cell sizes from a standard library, which may lead to suboptimal solutions. And sometime rounding may even cause timing violations or introduce extra area or power cost in discretization. Then some researchers use the continuous solution as a starting point, and then apply some methods to snap the continuous solution to discrete sizes. Shah et al. [23] solved discrete sizing problems using a self-snapping continuous formulations. Hu et al. [24] proposed a continuous-solution-guided and dynamic programming like approach. Rahman et al. [25] developed a branch-and-bound algorithm that maps the continuous sizes to the discrete sizes.

More recently, researchers tend to solve discrete sizing problem directly, rather than starting with continuous solutions. But it is hard to solve the combinatorial optimization problem as it is proved to be NP-hard by [26]. Chan [27] proposed an exhaustive search method, which is hard to afford for large size circuit. More recently, Held [28] presented a fast gate sizing heuristic that uses local slew gradient to estimate slew targets and minimize the circuit delay.

Greedy algorithms are widely used in previous works [29, 30, 31, 32, 33], which selects the optimal solution at each step. Coudert [34] proposed a greedy randomization based greedy method solving delay or power optimization. However, greedy algorithms may be trapped by local optimum and sometimes fail to reach the global optimum. Then Coudert et al. [35] tried to stay away from the infeasible region and avoid local minimum by applying global sizing techniques.

Dynamic programming is also quite useful in solving gate sizing problems [24, 27, 36]. Liu and Hu [36] presented a dynamic-programming-like searching method which iteratively improve solutions using bi-directional searching.

Lagrangian relaxation is widely used in discrete gate sizing [25, 36, 37, 38, 39, 40, 41] as well as continuous gate sizing[10, 11, 12, 13]. Lagrangian relaxation is a technique that combines the objective function with the constraints by multiplying Lagnrange multipliers and solves the subproblem and dual problem instead of solving the original problem directly. Lagrangian relaxation is flexible in solving area, delay or power minimization by solving. It is easy to implement and is proved to have optimal solutions with convex [10]. But Lagrangian relaxation also has its own limitation that it is limited to simply timing models, and may not solve complicated timing models.

In addition, there are some other techniques that have been explored by previous researchers. [17] used a linear program to minimize circuit power. [42] introduced the leakage power problem as a nonlinear mathematical program and used conjugate gradient method to solve.

As we can see that, the sizing problem has been well studied as researchers start with continuous sizing problems, and now work on discrete problems. And the gate model has been evolved from simply RC model to complex posynomial model, then to current lookup tables from standard cell libraries. But there is still a lot of room for further improvement, so we intend to solve the discrete gate sizing problem with real complicated gate models, and we may want to deal with extra slew and load capacitance constraints. And also, we intend to find methods that are more versatile, reliable and efficient, compared with all the existing works.

1.4 Outlines

This dissertation presents the discrete gate sizing methodology with circuit optimization of delay, area and power consumption.

The primary contributions of this dissertation are as follows: (i) We propose a series of methods which solves the discrete gate sizing problem concerning different optimization objectives, including delay, area and power consumption. (ii) Compared with other existing methods, our methods are faster as it applies the effective local evaluation in searching for cell replacement, and it is able to avoid the local minima. (iii) Our methods are compatible with modern gate cell models, such as lookup table from standard cell library. And it is able to handle complicated cell constraints, such as input/output slew constraint or

maximum output load capacitance constraint, which other methods may be limited with. (iv). We modified the oversimplified Elmore delay model with the features of the modern gate model so that it can be used to solve the discrete gate sizing problem with its accuracy and simplicity. Since the modified Elmore delay model works for gate sizing problems, other existing works based on Elmore delay may also be valid for library-cell-based problems.

In Chapter 1, we review the gate sizing problem with different optimization goals, and we see how the gate delay models evolve with the development of VLSI industry, which motivates us to dive into the problem. And we also find the limitations of previous research work, which give us a lot of challenges in finding better solutions to the problem.

In Chapter 2, we present a fast and efficient method that solves discrete gate sizing problem with delay minimization. In this work, when searching for the cell replacement, we apply the local delay difference in evaluating delay cost instead of global delay cost evaluation. The local delay difference greatly reduces the search space and improves the computational complexity. And more importantly, the method is able to avoid the local minima and reach the highly satisfactory global solution.

Then in Chapter 3, we present an algorithm of discrete gate sizing for area minimization with delay constraint. This method also applies the local delay difference in evaluating the delay cost when searching for cell replacement, which is efficient in runtime. But we modified some details to refine it for better precision. We apply the algorithm to further reduce the area while maintaining the minimum delay after applying the discrete gate selection method introduced in Chapter 2. Thus, we are able to find a solution that has the minimum circuit delay and optimal circuit area.

In Chapter 4, we present an algorithm that solves the gate sizing problem using lookup tables from standard cell library instead of Elmore delay model. The algorithm comes with two phases that does delay minimization and power optimization in one process. In addition, the algorithm is able to handle input slew constraints and output load capacitance constraints, which makes it more robust and versatile.

In Chapter 5, we propose a new gate model based on the classic Elmore delay model. The new delay model is able to characterize the timing features of the lookup tables from standard cell library. And with continuous Lagrangian relaxation, we are able to solve the gate sizing problem with delay minimization using the new model, and it is proved to provide good discrete solution after we apply refined nearest rounding method to discretization.

At last, in Chapter 6, we summarize the dissertation and point out the directions for future work.

Chapter 2

Delay Minimization for Discrete Gate Sizing Using Effective Local Delay Measurement and Cell Adjustment

Gate sizing problems has been critical to circuit design. However, previous researchers are more focused on solving continuous gate sizing problem, which is unrealistic in the real industry. And discretizing continuous solutions to discrete cell sizes usually lead to suboptimal solution. So in this chapter, we present an optimization algorithm to solve the discrete gate sizing problems. The algorithm is able to minimize the circuit delay based on effective local delay measurement and cell assignment, which is more efficient and faster compared with other exhaustive global searching methods. First our algorithm uses the continuous sizing solution as a starting point, and performs the local search of the gate cells on the critical path to gradually reduce the delay, and it continuous to perform the cell adjustment until no improvement. The experiments proved that our method is able to avoid being trapped by the local optima, and reach the highly satisfactory solution eventually. And experiments show that our method is within 4.9% more than the solution of the optimal continuous solution based on Lagrangian relaxation, which is ideal in theory but unrealistic in real. And it improves the circuit delay by 7.5% compared with nearest rounding.

2.1 Introduction

Generally speaking, gate sizing problem is to select proper sizes for each gate in the circuit for best performance. The circuit delay, layout area and power consumption may be varied by changing the gate sizes. And it is one of the most efficient and power techniques for the performance optimization of circuit design. And with the development of the manufacture process and the shrinking of transistor dimension, there is an increasing urge for efficient methodologies to satisfy the need of the modern design requirements. Researchers have been working on the gate sizing problem in the past a few decades. Initially, people were focusing on solving continuous sizing problems [8, 10, 17, 44, 45]. A fundamental work about continuous gate sizing problem is proposed by Chen et al. [10], who presented applied Lagrangian relaxation (LR) to solve the problem. Lagrangian relaxation is a useful technique to solve the sizing problem, since it can also be applied to handle certain statistical timing model [47] and sequential circuit [11, 48]. And the global continuous optimal solution is guaranteed with convex delay/area/power models [10, 13, 11, 36, 19].

As the continuous sizing problems that the circuit sizes can be any value in the continuous phrase, it is apparently untrue and unrealistic in real life. Hence, current researchers have shifted to solve discrete size problems with discrete cell library information. Therefore, it is our main concern to develop efficient algorithms that can handle discrete gate sizing problems based on standard cell library to achieve the best circuit performance.

Then people started looking for methods to discretize the continuous solutions to discrete sizes. Nearest Rounding [47, 48] is one of the most simply discretizing techniques,

which obtains an optimal continuous solution first, and then rounds the size of each gate to its nearest available discrete size. Though nearest rounding is simple to implement, it usually introduces large increment of delay and lead to suboptimal solutions after direct discretion, which leaves us considerable room for further improvement [24].

Among all the existing works, most of them are focused on optimizing power consumption or area. For example, in [48], linear programming and some heuristics are used to solve the area optimization problem under double-sided delay constraints, but it does not address the problem of delay minimization. Hu [24] proposed a discrete gate sizing algorithm for area optimization, which uses a dynamic programming approach and reports good results. However, it is very difficult to apply their search space pruning technique to the problem of delay minimization. Nguyen et al. [54] used linear programming to minimize dynamic and static power. Livramento et al. [39] used a state-of-the-art method to minimize leakage power with delay constraint. Liu et al. [36] proposed a systematic combinatorial approach with threshold voltage assignment.

Since there is limited number of existing works for solving discrete sizing problems with delay minimization, we intend to focus on minimizing circuit delay as the first priority. Constraint free delay minimization, which is addressed in this chapter, is very important but rarely studied in literature; only few researchers [10, 34, 40] have performed work on this, but they are either limited to continuous sizing [10], outdated delay model [34], or much more complicated approach than our proposed one [40]. Our approach is able to effectively limit the search space to.

In this chapter, we have the following contributions: we propose an approach to solve the discrete gate sizing problem with delay minimization, which few previous researchers

have done before. And we propose a few strategies to reduce the computational complexity and search space. First, the approach is a critical path based method, which focuses on reducing the critical path delay as the critical path delay determines the maximum delay of the entire circuit. Then we only consider replacing the current cells with those whose sizes are either one size larger or smaller than the current sizes. And most importantly, when evaluating the delay cost of cell replacement, we apply the local delay difference instead of global calculation, which greatly improves the efficiency. And it is able avoid the local minima by the "inaccurate" local delay calculation.

The rest of this chapter is organized as follows. In Section 2.2, we first introduce the notations used in the chapter and then formulate the delay minimization problem. In Section 2.3, we present the delay minimization algorithm with cell replacement. And in Section 2.4, we propose the local delay difference to measure the delay cost when selecting cell replacement. Then in Section 2.5, we show the flexibility of our algorithm as it also works with other modern gate models. The experimental results are shown in Section 2.6. And a summary of this chapter is presented in Section 2.7.

2.2 Problem Formulation

In this section, we first define the terminology and notations that we use in this chapter, and then formulate the problem to be discussed.

For the sizing problem, we focus on the combinational circuit only in this work. We model a combinational logic circuit as a directed acyclic timing graph $G(V, E)$, where V is the set of individual gates and E is the set of wires connecting the gates. In a circuit with n sizable gates, a logic gate with index i is represented as $v_i \in V$, where $1 \leq i \leq n$. Each wire connecting between nodes v_i and v_j is represented as directed edge $(v_i, v_j) \in E$, which propagates from gate v_i to gate v_j.

The set of primary input gates is represented as PI. For all the gates that are directly connected to the output loads, we add a pseudo output gate to connect them together, and the pseudo output gate is represented as PO. We use reverse topological order to label all the gates such that if gate v_i is connected to the output pin of gate v_j, then i is less than j. Suppose we apply reverse topological sorting to a circuit with s primary input gates, n sizable gates and t output loads, the pseudo output gate should be labeled as 0, the sizable gates are supposed to be labeled from 1 to n, as gates labeled from 1 to t as connected to the pseudo output gate, and the pseudo primary input gates are supposed to be labeled from $n + 1$ to $n + s$.

Figure 2. 1 shows an example circuit with 5 primary inputs, 6 sizable gates and 2 output loads. And all the primary inputs and sizable gates are labeled in reverse topological order. And the pseudo output gate PO is labeled as v_0, the sizable gates are labeled from v_1 to v_6, and the primary inputs are labeled from v_7 to v_{11}.

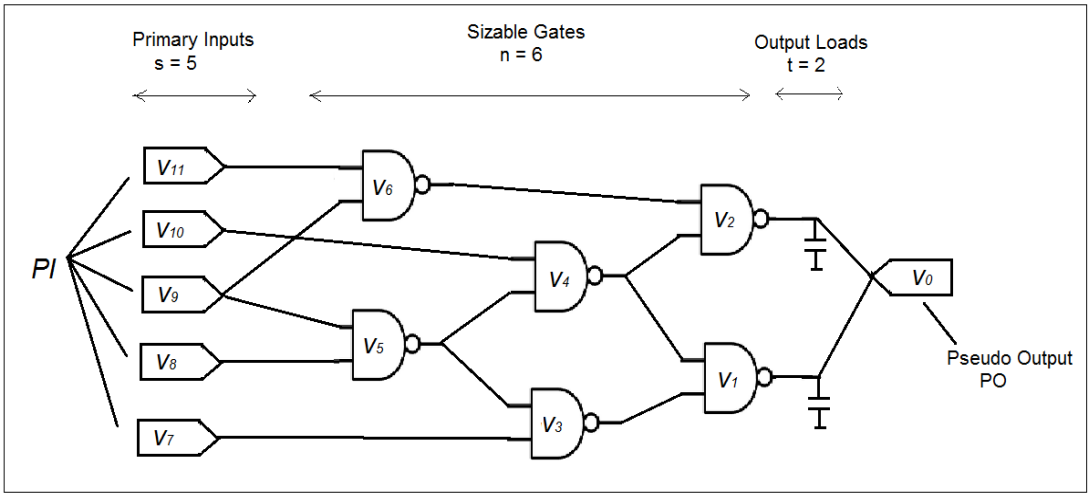

Figure 2. 1. An example circuit with gates labeled in reverse topological sorting order

For simplicity, we only discuss the sizing of combinational gates in this chapter, and leaving alone the wires, which is easy to be further refined with our algorithm. In a circuit, critical path is defined as the path with the largest delay among all the gate paths in the circuit. And there may be a few critical paths in a circuit. Apparently, the critical path delay determines the delay of the entire circuit.

Each gate $v_i \in V$ has multiple implementation options from the standard cell library, and each implementation instance is called a *cell*. So we use S_i to represent the set of all the available cells of gate v_i. And for each $v_i \in V$, it comes with a cell size $x_i \in S_i$. The minimum cell size is called the *lower bound*, and the maximum cell size is called the *upper bound*. And we use L_i and U_i to represent the lower bound and the upper bound for the cell size, as $L_i \leq x_i \leq U_i$. Then for a circuit with n sizable gates, we use a vector $X = (x_1, x_2, \dots, x_n)$ as an n-tuple vector to represent the discrete sizes of all sizable gates, where each x_i or $X(i)$ represents a discrete size for gate v_i and x_i is referred to as the i–th

component of X. X is simply referred as a *solution point* in many places of the following discussions.

With all the notations stated, we now formulate the discrete gate sizing problem of delay minimization subject to gate size restrictions as follows: given a circuit and cell library, find an optimal solution $X^* = (x_1^*, x_2^*, \dots, x_n^*)$ such that the worst circuit delay or the maximum arrival time of the output a_o is minimized:

$$Problem: \quad Find\ X^* = (x_1^*, x_2^*, \dots, x_n^*)\ that\ Minimizes\ a_O$$

$$subject\ to: \ a_j \le a_o, \forall v_j \in input(v_i), \forall v_i \in PO$$

$$a_j + D_i \le a_i, \forall v_j \in input(v_i), \forall v_i \in V$$

$$D_i \le a_i, \forall v_i \in PI$$

$$L_i \le x_i \le U_i$$

$$x_i \in S_i, \forall v_i \in V, i = 1,2, \dots, n$$

In the above equations, $a_i\ and\ D_i$ refer to the arrival time and delay of gate v_i, and a_o conrresponds to the maximum arrival time at the output. And $input(v_i)$ and $output(v_i)$ corresponds to all the gates that are directly connected to the inputs and outputs of gate v_i. And for the convenience of explanation and discussion, the Elmore delay model is employed for the timing analysis, and as we will show in section 2.5, our algorithm can also be applied with more accurate models, and it can also deal with the sizing of wires.

2.3 Delay Minimization with Gate Cell Replacement

Constraint free delay minimization algorithm based on Lagrangian relaxation from [10] provides an optimal delay solution in continuous gate sizing field. Though it usually introduces extra delay cost and becomes suboptimal when it is discretized to discrete phase by nearest rounding, it can be a very good solution as a starting point. So our algorithm will first apply nearest rounding to the continuous solution, and we then do the cell selection and adjustment based on the initial solution.

As a combinatorial problem, it is hard to solve discrete gate sizing problem efficiently. So to improve efficiency and eliminate unnecessary calculations, our priority is to resize those gates which will reduce the circuit delay most efficiently. Since the maximum circuit delay is determined by the delay along the critical path, we only focus on resizing the gates along the critical path.

Our approach begins with the continuous optimal solution, and we focus on resizing the gates on the critical path, and the set of all the gates along the critical path is called the *resizable gate set*. Then for each gate in the resizable gate set, we define the available cells from the cell library for adjustment, and measure their cost on the circuit delay if they are to replace the current cell. For the one which will reduce the delay for the most, we will replace the current cell with the delay minimizing cell. And we keep exploring the gates along the critical path and replace them with better cells until the delay cannot be further reduced along the critical path. Note that the critical path may be changed after the adjustment is done, so once we have fully adjusted one critical path, we will move to the updated critical path and repeat adjusting the new critical path gates. We will keep adjusting all the critical paths in the circuit until all the gates in the circuit are optimally

sized and the circuit delay cannot be further reduced. Thus, we find a solution point for solving the discrete delay minimization problem.

We outlined the discrete gate sizing algorithm with delay minimization in Figure 2. 2.

Algorithm 1: Discrete Gate Sizing Algorithm with Delay Minimization

Input: Circuit and Library Cell Information

Output: Solution X^* which minimizes the circuit delay

1. Solve continuous delay minimization problem using Lagrangian relaxation [10]
2. Discretize continuous solution using nearest rounding
3. **Repeat**
4. Update circuit timing information
5. Update set of critical path gates
6. Call Algorithm 2 to minimize delay on critical path
7. **Until** no further improvement or stopping criterion is met
8. Record solution X^* as the best solution of all iterations
9. Return X^*

Figure 2. 2. Pseudo code for Discrete Gate Sizing Algorithm with Delay Minimization

However, it is not an easy task to measure the delay cost for each available cell and determine which gate to be resized, as measuring the delay cost globally takes too much time and it is too expensive to afford. Thus, we propose a term called *local delay difference* to measure the delay cost locally, which largely reduces computational complexity and improves efficiency. In the following section, we will describe how to locally measure delay cost in detail.

2.4 Measuring Delay Cost by Local Delay Difference

In this section, we are to describe how to use local delay different to measure the delay cost when adjusting the gate sizes along the critical path. First we define some basic terms.

A. Neighboring Solution Set

In order to narrow down the search space, instead of investigating all the available standard cells for each gate, we only process the cells that are either one size smaller or one size larger than the current cell size x_i . We use x_l and x_h to represent the two neighboring cells, as $x_l \leq x_i \leq x_h$. And x_l and x_h are bounded by the lower bound and upper bound of the cell sizes, as $L_i \leq x_l \leq x_i \leq x_h \leq U_i$. And if $x_i = L_i$, $x_l = x_i$; if $x_i = U_i$, $x_h = x_i$.

In discrete space, the solutions are combinatorial. Even with the narrowed-down solution space, it is still considered exponential time for testing cases with all the solutions. In order to make our optimization process within linear time and not to be trapped by the local optima, we will only change exactly one gate cell at a time in this work. If we change the size of gate v_i from x_i to x_l or x_h, and keep all the other gates unchanged, we call it a *resize* of gate v_i. Hence, we define all the solution points that could be at after any single resize at solution point X as its *neighboring solution*. Therefore, for each solution X, the set of all the neighboring solutions of X is defined as vector :

$$N(X, x_i') = \left\{ x_j \middle| \begin{matrix} x_j = x_i' \text{ is either only size smaller or larger than } X(i) & \text{if } j = i \\ x_j = X(i) & \text{if } j \neq i \end{matrix} \right\},$$

Where $1 \leq i \leq n$.

B. Local Delay Difference

The critical path delay with size solution X is denoted as $D(X)$. Then among all the neighboring vectors $N(X, x_i'), where\ 1 \leq i \leq n$, their delays are noted as $D(N(X, x_i'))$. Then for each $N(X, x_i')$, the delay difference is denoted as $\Delta D(N(X, x_i')) = D(X) - D(N(X, x_i'))$, which is defined as *the delay cost of* $N(X, x_i')$. Our problem is to find the $N(X, x_i')$ with the maximum delay difference at solution point X, which is denoted as $\Delta D_{max}(X, x_i') = \max\{\Delta D\big(N(X, x_i')\big), \forall x_i'\}$.

To find the $N(X, x_i')$ with $\Delta D_{max}(X, x_i')$, we need to resize every gate along the critical path respectively and calculate $\Delta D(N(X, x_i'))$, which requires high computational cost. To reduce the computational complexity, we introduce a target-path-based method to obtain the discrete delay difference, which only considers the gates directly connected to the gate being resized at the time. Since the method only evaluates the gate delay locally rather than globally, we call the obtained delay difference as *local delay difference*, which will be further discussed in part D of delay cost measurement.

C. Target Local Gate Set

Then we define the *Target Local Gate Set*, which is greatly helpful in determining the local delay difference. When considering measuring the resizing effect on a certain gate v_i, we only take two kinds of gates into consideration. First kind of gates are those gates that directly drive the gate v_i, which we call as *predecessor gates*. And second kind of gate is the one which is driven by v_i and also on the critical path, which we call it as *direct successor gate*.

Figure 2. 3 shows an example circuit whose critical path is shown in red ink. When we evaluate the delay cost of resizing the gate v_i, we need to take predecessor gates v_{j0} and v_{j1}, the direct successor gate on the critical path v_k, and v_i itself into consideration. So the target local gates set include the predecessor gates, the gate being resized, and the successor gate.

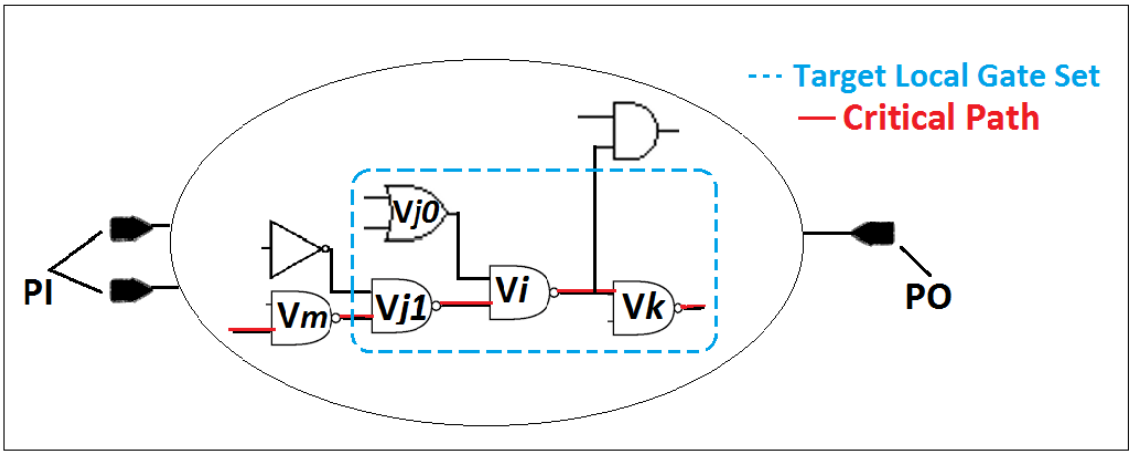

Figure 2. 3. Illustration of Target Local Gate Set

Then considering the gate v_i on the critical path in Figure 2. 3, assume its current cell size is x_i , gate capacitance is c_i and arrival time is a_i, and we will see how to calculate the delay cost measurement of replacing it with a new cell x_i' with gate capacitance c_i' and gate resistance R_i'.

We use Elmore delay to evaluate circuit delay. With Elmore delay, for gate v_i, its gate delay D_i is:

$$D_i = R_i \times OC_i \tag{2. 1}$$

where R_i is the gate resistor of v_i, and OC_i is the output load capacitance of v_i.

If we replace gate v_i with a new cell x_i', obviously it will change the capacitance, delay, transition time and arrival time of itself, and also it will change the delay and arrival time of its predecessor gates. Besides, it will further affect its direct successor gate since the new arrival time will affect its delay and arrival time, too. So we would like to discuss the details in the following equations:

1. For all the predecessor gates $v_j \, where \, v_j \in input(v_i)$, their new delay d_j' and arrival time a_j' are:

$$d_j' = R_j \times (OC_j + c_i' - c_i) \tag{2. 2}$$

$$a_j' = a_j + d_j' - d_j \tag{2. 3}$$

2. For gate v_i itself, its new delay d_i' and arrival time a_i' are:

$$d_i' = \ R_i' \times OC_i \tag{2. 4}$$

$$a_i' = max(\underset{v_j \in input(v_i)}{a_j'}) + d_i' \tag{2. 5}$$

3. Then for the director successor gate v_k, its new delay d_k will not change as its own gate size is the same, but its arrival time a_k' may change to:

$$a_k' = \begin{cases} a_i' + d_k \, , & v_k \notin PO \\ a_i', & v_k \in PO \end{cases} \tag{2. 6}$$

With all the equations stated, we now evaluate the delay cost of resizing gate v_i by the arrival time difference of its direct successor gate. Thus, we define the equation for the local delay difference as:

$$\Delta D(N(X, x_i')) = \underset{v_k \in output(v_i)}{a_k - a_k'} \quad (2.7)$$

With the cost measure explained, we listed the pseudo code of local delay difference algorithm in Figure 2. 4. In the algorithm, we iteratively examine the gates along the critical path, and for each gate, we explore the local delay difference of each neighboring cell, and find the one which will decrease the delay most and replace the gate with the new cell. And we keep doing the process until new further improvement.

We will show in our experiment in Section 2.5 that our method is able to climb out of local minima and converge to the best solution.

Algorithm 2: Local Delay Difference Algorithm

Input: Circuit with solution set X with critical path CP

Output: Solution X' which minimizes the delay on CP

1. Start with initial solution set X
2. **Repeat**
3. Foreach gate along the critical path, define the set of $N(X, x_i')$
4. Obtain Local Delay Difference $\Delta D(N(X, x_i'))$
5. Find the $N(X, x_i')$ with $\Delta D_{max}(N(X, x_i'))$
6. If $\Delta D_{max}(N(X, x_i')) > 0,\ X \leftarrow N(X, x_i')$
7. Update circuit timing information
8. **Until** $\Delta D_{max}(N(X, x_i')) \leq 0$
9. Return X' with minimum delay

Figure 2. 4. Pseudo code for Local Delay Difference Algorithm

2.5 More Accurate Gate Model

In Section 2.3, we use Elmore Delay model to illustrate our algorithm for simplicity. However, since Elmore delay is usually criticized for its inaccuracy, more accurate gate delay models are preferable for more demanding timing requirements. And modern gate models usually take input slews into consideration. Luckily, our approach is able to work with more accurate gate models. So in this section, we will show how to extend our algorithm to other modern delay models. Since discrete gate sizing problems are combinatorial, modern gate delay models can be easily adopted into our algorithm.

With more accurate gate model, we need to modify the equations in Section 2.3. We generalized the gate delay of v_i as D_i as a function of its size, output load capacitance, and input slew, denoted as:

$$d_i = f_i(x_i, OC_i, Input\_Slew_i) \tag{2. 8}$$

Then we will repeat the equations in Section 2.3 and see how it will affect the local delay difference. Note that the input slew is also a function of size, output load capacitance and other parameters, the input slew of a gate will change its value if the sizes of its driving gates change. We now review the delay and arrival time of the gates in the target local gate set:

1. For all the predecessor gates $v_j \; where \; v_j \in input(v_i)$, their new delay d_j' and arrival time a_j' are:

$$d_j' = f_j(x_j, OC_i', Input\_Slew_i) \tag{2. 9}$$

$$a_j' = a_j + d_j' - d_j \tag{2. 10}$$

2. For gate v_i itself, its new delay d_i' and arrival time a_i' are:

$$d_i' = f_i(x_i, OC_i, Input\_Slew_i') \tag{2. 11}$$

$$a_i' = max(\underset{v_j \in input(v_i)}{a_j'}) + d_i' \tag{2. 12}$$

3. Then for the director successor gate v_k, its new delay d_k changes as its input slew changes with x_i, but its arrival time a_k' may change to:

$$d_k' = f_k(x_k, OC_k, Input\_Slew_k') \tag{2. 13}$$

$$a_k' = \begin{cases} a_i' + d_k', & v_k \notin PO \\ a_i', & v_k \in PO \end{cases} \tag{2. 14}$$

Then with Eq. (2. 7): $\Delta D(N(X)) = \underset{v_k \in output(v_i)}{a_k - a_k'}$ and all the equations above, we now can perform local delay difference for more accurate delay models. Hence, our algorithm can still perform with different delay models.

2.6 Experimental Results

The discrete gate sizing algorithm is implemented and performed on a 3.4 GHz Intel i-7 Core computer. We use ISCAS ’85 benchmark circuits as test cases, which are commonly used for circuit analysis and optimization (see [51, 52]). And we use a 22-nm technology cell library, and the cell library has 10 sizes for each type of gate. The 10 sizes are 1×, 2×, 3×, 4×, 6×, 8×, 16×, 32×, 64×, 128× of the minimum size with respect to each cell type. The unit capacitance and resistance of each type of gate is provided by the cell library.

First on concerning the delay performance, we compare the experimental results of our algorithm with the solutions of Continuous Langrangian Relaxation Algorithm [10] and nearest rounding. Nearest rounding algorithm obtains discrete solutions by rounding the continuous size solution to the nearest discrete size directly. Though Continuous Lagrangian Relaxation is proved to obtain the minimum continous circuit delay, it may lead to suboptimal solution when discetized to discrete solution.

In Table 2. 1, we first list the minimum circuit delay from Continuous LR Solution. Apparently, the continuous solution gives the best circuit delay in theory, which is unrealistic in real life. Then from the circuit delay data of the nearest rounding part, we can see that nearest rounding solution usually introduces a large amout of delay cost, as it has an average of 14.0% more than the circuit delay of continuous LR solution, which is not ideal for real circuit implementation. And our algorithm of discrete gate sizing is able to reduce the delay by around 7.5% compared with nearest rounding, and it’s no more than averagely 4.9% circuit delay increment from the continuous LR solutions, which is very close to the optimal continuous solution.

To better illustrate how our algorithm reduces the delay as the iterations go on, Figure 2. 5 and Figure 2. 6 show the circuit delay sequence of circuit c2670 and circuit c3540 from continuous solution with Lagrangian relaxation via nearest rounding to discrete sizing algorithm. For c2670, we omit iterations 34 to 795 for illustration purpose. As we can see in the figure, after nearest rounding, circuit delay goes up for approximately 20.5%. And for c3540, we omit iterations 31 to 572. And the nearest rounding will introduce 13.6% circuit

Table 2. 1. Minimum Circuit Delay comparison of discrete sizing algorithm (DS) with continuous solution (CS), nearest rounding (NR).
Delays are in *ps*.

	Circuit	# of gates	Circuit Delay			Delay Comparison (%)		
			Continuous Solution (CS)	Nearest Rounding (NR)	Discrete Solution (DS)	DS vs. CS	DS vs. NR	NR vs. CS
1	c432	160	104.0	114.9	108.6	4.4%	-5.5%	10.5%
2	c499	202	86.0	86.5	86.1	0.1%	-0.5%	0.6%
3	c880	383	83.0	91.2	86.5	4.2%	-5.2%	9.9%
4	c1355	546	109.0	111.1	111.1	1.9%	0.0%	1.9%
5	c1908	880	117.0	144.3	121.6	3.9%	15.7%	23.3%
6	c2670	1193	114.0	137.4	127.3	11.7%	-7.4%	20.5%
7	c3540	1669	148.0	168.2	158.5	7.1%	-5.8%	13.6%
8	c5315	2307	137.0	151.2	143.9	5.0%	-4.8%	10.4%
9	c6288	2416	386.6	519.2	407.4	5.4%	-21.5%	34.3%
10	c7552	3512	135.4	156.1	143.0	5.6%	-8.4%	15.3%
Avg.		1327	142.0	168.0	149.4	4.9%	-7.5%	14.0%

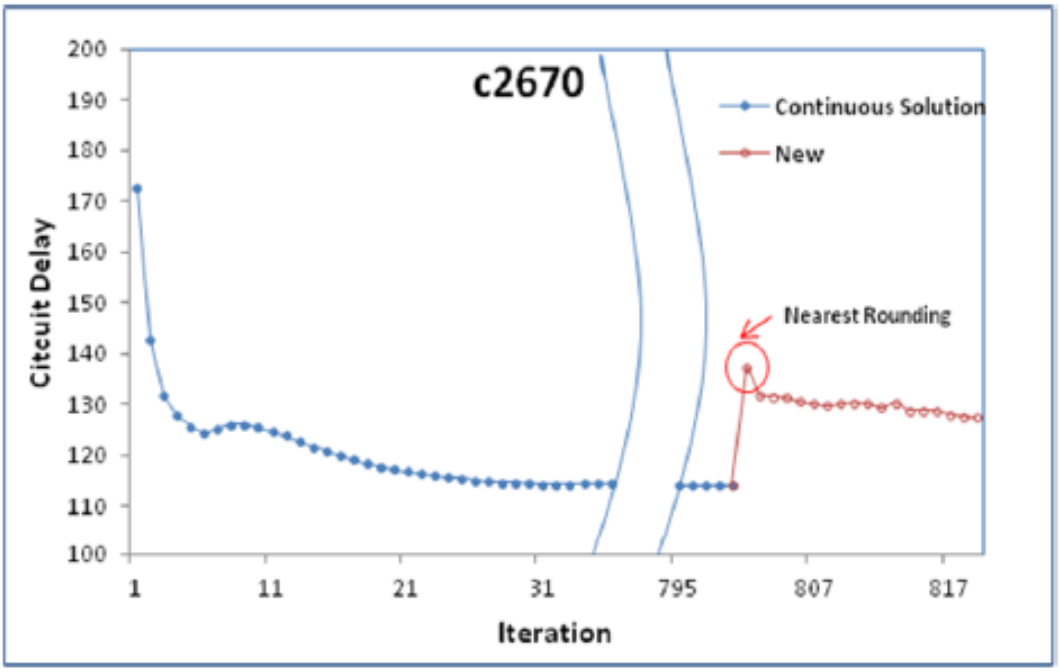

Figure 2. 5. Circuit delay sequence of c2670, iterations 34 to 795 are eliminated. (Continuous Solution -> Nearest Rounding -> Discrete Sizing)

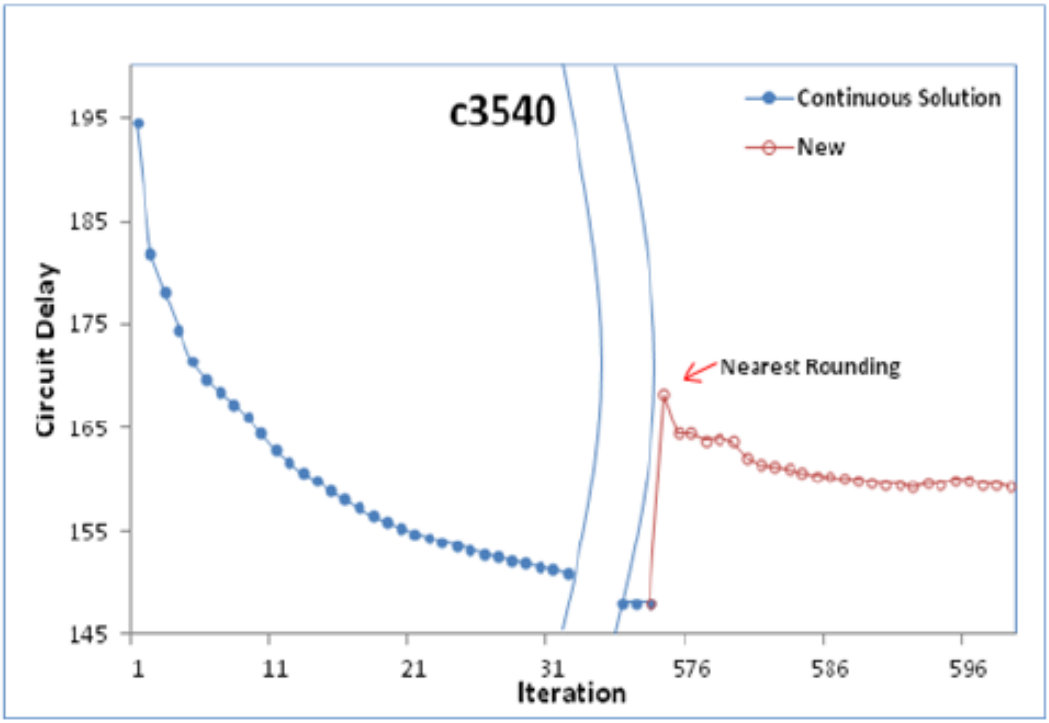

Figure 2. 6. Circuit delay sequence of c3540, iterations 31 to 572 are eliminated. (Continuous Solution -> Nearest Rounding -> Discrete Sizing)

delay as shown in the figure. And during the discrete sizing process, the circuit delay may increase at some iterations, which is caused by the "inaccurate" delay cost measurement of local delay difference. And it is because of the inaccuracy which allows the delay not to be trapped by the local minima and gradually reach the global optima in the end.

To compare the runtime performance, Table 2. 2 shows the runtime comparison of our algorithm (DS) with continuous solution (CS). Since our algorithm needs the continuous solution as the starting point, the total runtime is the sum of CS and DS. And it's displayed that the runtime of DS is within average of 36.74% more than CS.

Table 2. 2. Runtime Comparison between Discrete Solution and Continuous Solution. *Runtimes* are in *seconds (s).*

	Circuit	# of gates	Runtime		Comparison
			Continuous Solution (CS)	Discrete Solution (DS)	DS vs. CS
1	c432	160	0.124	0.022	17.74%
2	c499	202	0.064	0.061	95.31%
3	c880	383	0.204	0.080	39.22%
4	c1355	546	0.088	0.110	125.00%
5	c1908	880	0.204	0.082	40.20%
6	c2670	1193	0.257	0.055	21.40%
7	c3540	1669	1.275	0.650	50.98%
8	c5315	2307	2.553	0.179	7.01%
9	c6288	2416	7.396	1.144	15.47%
10	c7552	3512	1.767	2.735	154.78%
Avg.		1327	1.393	0.512	36.74%

2.7 Summary

We have presented an approach to the discrete sizing problem of delay minimization. Our algorithm is able to reach highly satisfactory solution in a fast fashion as it largely reduces search space by only considering resizing the gates on the critical path and allowing a small amount of resizable cells. Most importantly, with local delay difference, which is used to measure the resizing delay cost, it is able to avoid local minima and finally reach the highly satisfactory global optimum as the iteration goes on.

We have experimented our algorithm on 10 circuits of ISCAS ’85 benchmarks, and the experimental results show that this approach can handle gate sizing problem fast and efficiently, with a 7.5% delay improvement compared with nearest rounding method, which is commonly used. And the average runtime of our algorithm is within only 36.74% more than that of the continuous Lagrangian relaxation sizing process.

Chapter 3

Post Area Optimization with Minimum Circuit Delay Constraint

In this chapter, we present an approach which reduces circuit area in a quick fashion while maintaining its current circuit delay. We may apply this area optimization method after we obtain the discrete solution with minimum circuit delay using the method introduced in Chapter 2. In this way, we will have a solution with minimum circuit delay and optimized circuit area at the same time. Since this method is intended to be applied after running the delay minimization method, we call it as post area optimization.

The post area optimization method is an iterative heuristic method. It starts with the discrete solution with minimum circuit delay obtained from Chapter 2. To reduce the circuit area and not to violate the minimum delay constraint, we search for possible cells with smaller sizes for replacement. And we use the local delay difference to measure the delay cost when determining whether to replace a cell or not.

From our experiment, our method is able to reduce the area by 5.6% on the average and it is within 8.0% more than the optimal continuous solution.

3.1 Introduction

Gate sizing is an important part in circuit design as it can be used to improve the circuit performance. And since the gate sizing problems has been studied for several decades, many researchers have been working on the problem from different aspects. Thus, we will first review the existing gate sizing methods.

First of all, the existing gate sizing methods can be divided as continuous methods and discrete methods. The continuous methods take the gate sizes as continuous numbers, while the discrete methods solve the gate sizes as discrete numbers. Among the continuous methods [7, 9, 10, 12, 14, 19, 44], an early representative work is from Fishburn and Dunlop [8], who presented a sensitivity-based greedy heuristic to solve the continuous sizing problems. And another important work is from Chen et al [10], who solved the continuous sizing problem using Lagrangian relaxation, which is proved to gain the optimal solution.

Then among the discrete methods, Roy et al. [48] rounded the continuous solution to discrete solution. Some works [23, 24, 43] used the continuous solution as a starting point. And some other works [34, 36, 50, 54] solve the discrete gate sizing problems directly. Continuous methods usually depend on the mathematical methods to solve and are proved get optimal solution, but the continuous solutions are impractical in real life. Discrete solutions are hard to solve and most existing works rely on heuristic methods, and cannot guarantee optimality.

Then for the optimization goal, the existing gate sizing methods can be divided as area minimization, timing/delay minimization and power minimization. Among the works with area minimization [9, 10, 11, 24, 49], Sapatnekar et al. [9] used convex programming to

solve delay constrained area minimization problem. Hu et al. [24] introduces a continuous-solution-guided dynamic programming approach to solve cell-library-based design problems on minimizing circuit area under delay constraint. Then in [49], linear programming and some heuristics are used to solve the area optimization problem under double-sided delay constraints. Among the works of delay minimization [24, 25, 34, 36, 38, 40], Coudert [34] presented a discrete gate sizing algorithm to minimize the maximum circuit delay by maximizing the smallest slack. Liu and Hu [36] propose a method to select cells by iteratively relaxation and restoration. Among the works of power minimization [12, 25, 29 - 33, 50], Livramento et al. [50] presents an algorithm for leakage power minimization based on Lagrangian Relaxation.

In this chapter, we propose a method to optimize the circuit area while remaining the circuit delay. We apply the local delay difference from Chapter 2 when searching for available cell replacement.

The rest of this chapter is organized as follows. In Section 3.2, we first introduce the notations used in the chapter and then formulate the area optimization problem. In Section 3.3, we present the area optimization methodology with local searching for the cell replacement. Then we show the experimental results in Section 3.4. A summary of this chapter is presented in Section 3.5.

3.2 Problem Formulation

In this section, we will formulate the discrete gate sizing problem for area minimization under delay constraint. Before that, we would like to give a short introduction of the notations that we use in this chapter.

For the notations, we follow our circuit model defined in Chapter 2, where the set of gates is noted as V, and the set of wires that connect the gates is noted as E. And the circuit is modeled as a directed acyclic timing graph, noted as $G(V, E)$. The set of primary input gates is represented as PI, and the pseudo output gate is noted as PO. And in a circuit with n sizable gates, each individual gate with an index i is noted as gate $v_i \in V$, where $1 \leq i \leq n$. And all the gates are labeled in reverse topological sorting order. And each individual gate comes with a size x_i, gate delay d_i and arrival time a_i. And the arrival time of the pseudo output gate a_0 is the same as the circuit delay. And we use an n-tuple vector $X = (x_1, x_2, \dots, x_n)$ to represent the discrete sizes of all the gates in the circuit.

We state the discrete sizing problem with area minimization as follows: given a circuit and solution point X with its current circuit delay or arrival time a_0, find the optimal solution X^* that minimizes the total circuit area while maintaining its circuit delay. Hence, the problem can be formulated as following:

$$Problem: \quad Find\ X^* = (x_1^*, x_2^*, \dots, x_n^*)\ that\ Minimizes \sum_{i=1}^{n} x_i$$

$$subject\ to: \ a_j \leq a_o, \forall v_j \in input(v_i), \forall v_i \in PO$$

$$a_j + D_i \leq a_i, \forall v_j \in input(v_i), \forall v_i \in V$$

$$D_i \leq a_i, \forall v_i \in PI$$

$$L_i \leq x_i \leq U_i$$

$$x_i \in S_i, \forall v_i \in V, i = 1,2, \dots, n$$

3.3 Optimization Methodology

We intend to reduce the total circuit area by replacing certain gates with smaller cells while not violating the timing constraint. Now it becomes a critical problem since each gate may have a few options of smaller cells and it is hard to tell which ones to be replaced. And it may be computationally expensive if we recalculate the circuit delay when evaluating the delay cost of each cell. So to reduce the computational complexity, we follow the concept of local delay difference introduced in Chapter 2 to help us decide which cell to be replaced.

Before we describe our methodology, we would like to start with the definitions of some basic terms.

A. Available Cell Set

For the gate v_i with current size x_i, we only consider all the cells whose sizes are smaller than x_i. So we define the set of all the available cells as $A(X) = \{x_i' < x_i, \ | \forall v_i \in \ G, i = 1,2, \dots, n\}$.

B. Delay Cost Measurement

Again, we follow the analysis in Chapter 2 to evaluate the delay cost measurement, but since all the resizable cells are smaller than the current size, as $x_i' < x_i$, and their gate capacitance are also smaller, as $c_i' < c_i$, we can elaborate more features from those equations:

1. For all the predecessor gates $v_j where\ v_j \in input(v_i)$, their new delay d_j' and arrival time a_j' are:

$$d_j' = R_j \times (OC_j + c_i' - c_i) \tag{3.1}$$

$$a_j' = a_j + d_j' - d_j \tag{3.2}$$

$a_j' = a_j + d_j' - d_j \Rightarrow \Delta d_j = a_j' - a_j < 0$, so the arrival time of all the predecessor gates will be decreased, and will not affect the arrival time of those gates.

2. For gate v_i itself, its new delay d_i' and arrival time a_i' are:

$$d_i' = R_i' \times OC_i \tag{3.3}$$

$$a_i' = max(\underset{v_j \in input(v_i)}{a_j'}) + d_i' \tag{3.4}$$

Now let's take a look at $\Delta d_i = a_i' - a_i$. Concerning the value of Δd_i, we have the following three cases:

Case 1: $\Delta d_i = a_i' - a_i \leq 0$.

If the local delay difference is non-positive, it means that resizing the gate with a smaller cell will not increase the delay along the path, so we can definitely replace it with the new smaller cell.

Case 2: $\Delta d_i = a_i' - a_i > 0$ and gate v_i is on the critical path.

Apparently replacing it will increase the entire circuit delay, so we can't replace it and abort this option.

Case 3: $\Delta d_i = a_i' - a_i > 0$ but v_i is not on the critical path.

In this case, we need to further investigate the effect of the increased delay to the output gates of v_i.

Figure 3. 1 shows an example circuit with gates v_{i0}, v_{i1} and v_k, and gate v_{i1} and v_k are on the critical path. When considering replacing gate gates v_{i0} with a smaller size, we find its local delay difference $\Delta d_{i0} = a'_{i0} - a_{i0} > 0$. Since we can not determine whether it will increase the entire circuit delay, we need to continue investigate the effete on its output gate v_k. For gate v_k, its critical input may or may not change because of the replacement. If $a'_{i0} > a_{i1}$, the arrival time of gate v_k will be increased since its input arrival time has increased, thus we are not supposed to do this replacement. But for the case $a'_{i0} \leq a_{i1}$, then the arrival time of gate $v_k$will not be changed since it input arrival time does not change, thus we can do the replacement.

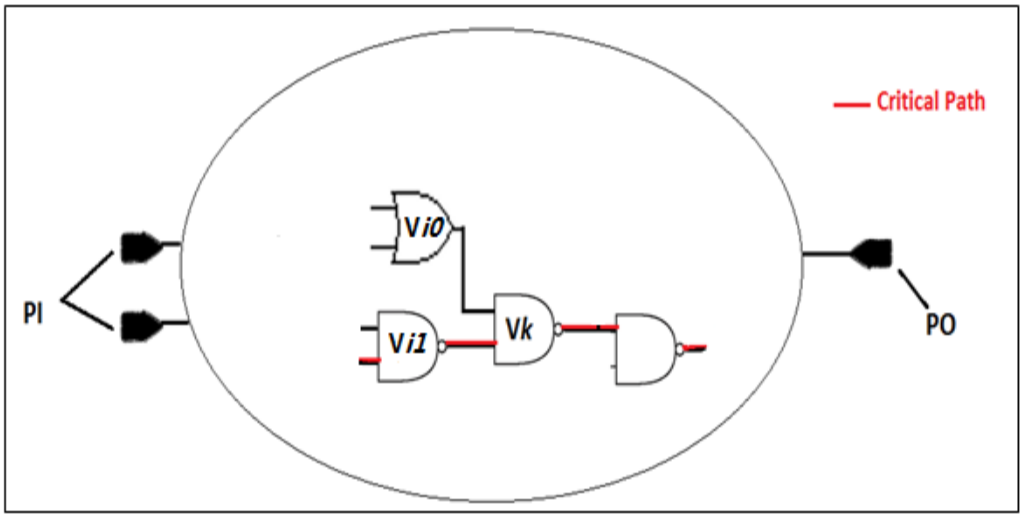

Figure 3. 1. An example circuit.

So for each gate $v_k, where\ v_k \in output(v_i)$, we simply calculate its new added delay $\Delta d_k = a_i + \Delta d_i - a_{CI(v_k)}, where\ a_{CI(v_k)}$ is the arrival time of critical input of v_{k}, as critical input is the input gate which has the largest arrival time. Then we can propagate the increased delay and continue the investigation.

We define all the three cases in a function called *Replaceable*. Function *Replaceable* returns true if we can replace a gate with a new cell. The pseudo code for the function is shown in Figure 3. 2.

Function Replaceable (gate v_i, delay differenc Δd_i)

Input: gate v_i and its gate delay difference Δd_i

Output: *True* if the circuit delay will not be increased with Δd_i. *False* if else.

```
1.  If (Δd_i ≤ 0)
2.      Return true.
3.  Else
4.      If (v_i is on Critical Path)
5.          Return false.
6.      Else
7.          Foreach gate v_k, where v_k ∈ output(v_i)
8.              Δd_k = a_i + Δd_i − a_CI(v_k)
9.              If (Replaceable(v_k, Δd_k) == false)
10.                 return false.
11.         Return true.
```

Figure 3. 2. Pseudo code for Function *Replaceable*

The Post Area Optimization algorithm is outlined in Figure 3. 3.

Algorithm 3: Post Area Optimization Algorithm

Input: Solution X with minimum circuit delay

Output: Solution X^* with same minimal circuit delay but optimized area

1. Start with initial solution set X
2. Repeat
3. Define the set of $A(X)$
4. Foreach gate v_i
5. Foreach cell x_i' in $A(X)$, start from minimum size
6. Obtain $\Delta d_i(A(X))$
7. If(Replaceable ($v_i, \Delta D_{max}(A(X))$)== *true*)
8. $X \leftarrow A(X)$, break
9. Update circuit timing information
10. Stop until no further improvement or stop requirement is met
11. Return $X^*$with minimum circuit delay and optimized area

Figure 3. 3. Pseudo code for Post Area Optimization algorithm

3.4 Experimental Results

We have implemented the post area optimization algorithm and performed the experiments on a 3.4 GHz Intel i-7 Core computer. We use ISCAS 85' benchmark for circuit optimization and performance analysis. And the 22-nm standard cell library is used as a discrete cell library. Each type of gates has ten different cells, and they are either $1\times$, $2\times, 3\times, 4\times, 6\times, 8\times, 16\times, 32\times, 64\times,$ or $128\times$ of the minimum size.

Table 3. 1. Minimum Circuit Area and Delay Comparison of Post Area Optimization (PAO) with Initial Discrete Solution (DS),and Continuous Solution (CS).
Delays are in *ps*.

	Circuit	# of gates	Initial Discrete Solution		Discrete Solution w/ PAO		Continuous Solution		Area Comparison	
			Area	Delay	Area	Delay	Area	Delay	PAO vs. DS	PAO vs. CS
1	c432	160	364.0	108.6	348	109.4	322.2	109.1	-4.4%	8.0%
2	c499	202	1241.0	86.1	1185	87.06	1021.2	86.2	-4.5%	16.0%
3	c880	383	1184.0	86.5	1098	87.31	974.4	85.1	-7.3%	12.7%
4	c1355	546	1960.0	111.1	1910	111.99	1807.4	110	-2.6%	5.7%
5	c1908	880	1969.0	121.6	1880	122.6	1816.5	121.7	-4.5%	3.5%
6	c2670	1193	2247.0	127.3	1961	128.3	1711.1	128.1	-12.7%	14.6%
7	c3540	1669	2724.0	158.5	2500	159.3	2297.8	158.1	-8.2%	8.8%
8	c5315	2307	3385.0	143.9	3300	144.8	3323.4	144	-2.5%	-0.7%
9	c6288	2416	3889.0	407.4	3685	408.4	3536.2	408.6	-5.2%	4.2%
10	c7552	3512	4936.0	143.0	4748	143.9	4415.5	142	-3.8%	7.5%
Avg.		1327	2389.9	149.4	2262	150.3	2122.6	149	-5.6%	8.0%

Table 3. 1 shows the area and delay of initial discrete solution, discrete solution with post area optimization (PAO) method applied, and continuous solution from area minimization with delay constraint based on Lagrangian relaxation [10]. In our experiments, we first apply the post area optimization algorithm to the discrete solution with minimum delay, which was introduced in Chapter 2. And we compared the area of PAO algorithm with the initial discrete solution. As shown in Table 3. 1, compared with the initial discrete solution, the circuit area after applying PAO is reduced by an average of 5.6%, ranging from 12.7% to 2.5%. Then to judge the efficacy of our algorithm, we compared our results with continuous solution which is derived from area minimization with delay constraint using Lagrangian relaxation [10]. And compared with the continuous solution, our algorithm is only within 8% more area in average.

The post area minimization method is to shrink the size of certain gates whose size decrement will not affect the circuit delay. To better illustrate this idea, the histogram of gate size percentage for before and after PAO are shown in Figure 3. 4 and Figure 3. 5. And we can see that after running PAO, the percentage of smaller cells after PAO is much more increased.

Table 3.2 shows the runtime comparison between the post area minimization algorithm and continuous area minimization algorithm with Lagrangian relaxation [10]. And the runtime of the two algorithms are very close as the runtime of PAO is 5.24% less on average.

Table 3.2 shows comparison between runtime of PAO and the number of gates. By performing a linear regression on the logarithm of the data in Figure 3. 6, we find the empirical runtime of PAO algorithm is about $O(n^{1.07})$.

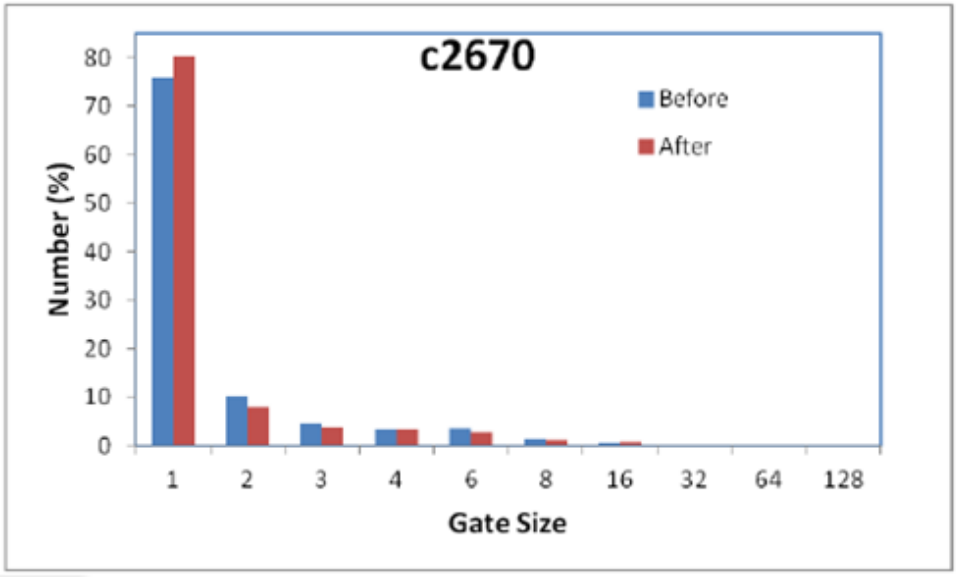

Figure 3. 4. Gate-Size percentage histogram for before and after PAO of c2670

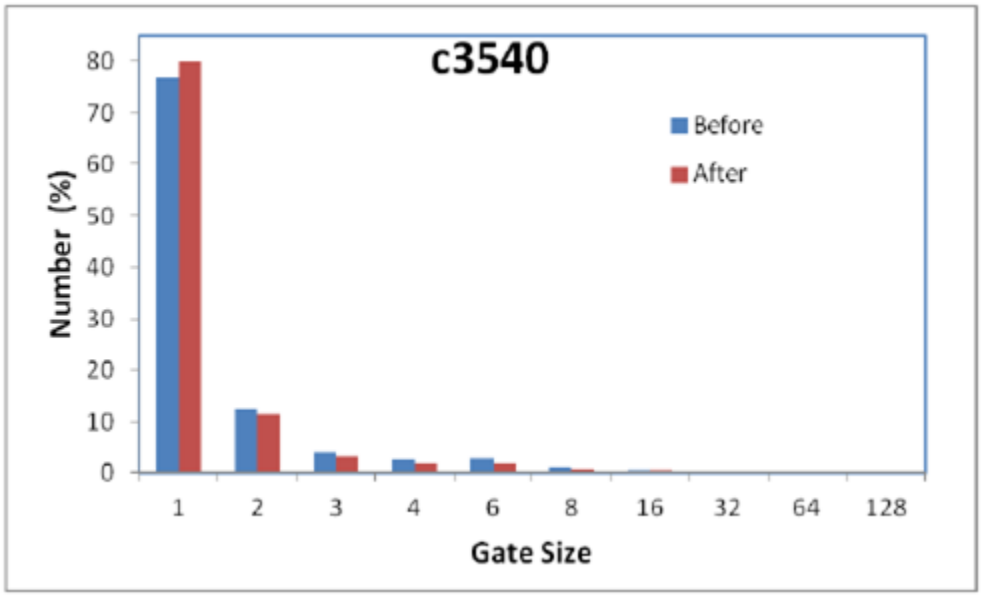

Figure 3. 5. Gate-Size percentage histogram for before and after PAO of c3540

Table 3.2. Runtime Comparison between Post Area Optimization (PAO) with Continuous Solution (CS). *Runtimes* are *seconds(s).*

	Circuit	# of gates	Runtime		Comparison
			PAO	Continuous Solution (CS)	PAO vs. CS
1	c432	160	0.058	0.238	-75.63%
2	c499	202	0.421	0.235	79.15%
3	c880	383	0.279	0.452	-38.27%
4	c1355	546	0.759	0.653	16.23%
5	c1908	880	1.193	1.999	-40.32%
6	c2670	1193	2.118	2.570	-17.59%
7	c3540	1669	4.126	3.044	35.55%
8	c5315	2307	7.985	6.839	16.76%
9	c6288	2416	11.327	10.323	9.73%
10	c7552	3512	17.665	22.117	-20.13%
Avg.		1327	4.593	4.847	-5.24%

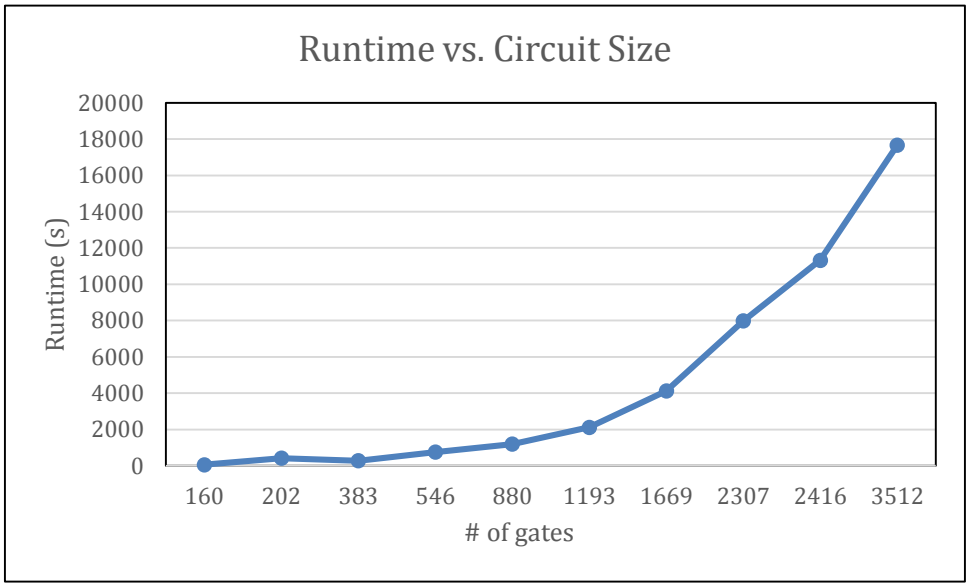

Figure 3. 6. The runtime of PAO algorithm vs. the number of gates

3.5 Summary

In this chapter, we intend to further reduce the circuit area after gaining the circuit solution with minimum delay using the method introduced in Chapter 2. Thus, we have presented a discrete gate sizing method which aims to minimize circuit area while remaining the minimum circuit delay. Our algorithm also applies the local delay difference introduced in Chapter 2 to evaluate the delay cost when selecting cell replacement, which is efficient as it largely reduces computational complexity.

We have experimented our area minimization algorithm on ISCAS ’85 benchmarks, and the experimental results show that this approach is able to reduce the area by 5.6% and it is within 8.0% more than the optimal continuous solution.

Chapter 4

Library-Cell-Based Discrete Gate Selection for Delay Minimization and Power Optimization

We present a fast and efficient critical-path-based heuristic for the discrete gate sizing problems, which, as a first objective, minimizes circuit delay and then, as a post-processing step, minimizes power consumption. Experimental data also exhibit a trend of linear time behavior for the proposed heuristic. The approach works with discrete cell library gates under the constraints of input/output transition time and load capacitance, as specified in the timing tables of the library cells. The search space is significantly reduced by considering the set of resizable neighboring solutions and gates on the critical path only. A non-greedy method is used to assess the effect of a specific gate sizing, which allows the proposed heuristic to effectively climb out of local optima, and eventually lead to highly optimized global solutions. Then we propose a post power optimization method to further minimize the power consumption while retaining the minimized circuit delay. The experimental results show that our algorithm is able to achieve very satisfactory solutions and it takes less than 1 second for a circuit with about 4000 gates.

4.1 Introduction

As the technology scales down and the density of transistors grows up, there is an increasing urge for efficient methodologies to better achieve the performance of modern very large scale integrated (VLSI) circuits. Thus, with the objective to optimize circuit performance, the gate sizing problem of selecting appropriate size for each gate has been a critical issue. Changing gate sizes in a circuit can significantly affect its delay, layout area and power consumption. Earlier researchers focused on continuous size solutions with a highly simplified delay model [8, 10, 11, 15, 48], however with the development of integrated circuit industry, developers are more concerned about selecting discrete gate sizes from standard cell library to optimize their circuit design [24, 28]. Therefore, it is of crucial importance to develop efficient algorithms, which can handle discrete gate sizing problems based on standard cell library to achieve the best circuit performance.

Although only discrete gate sizing is practical and what designers really need in the IC industry, it is very hard to solve the discrete problem considering its NP-hard complexity, as described by Li [26]. So previous researchers were more focusing on continuous sizing problems, assuming that the circuit sizes can be any value in the continuous phrase [8, 10, 44, 45, 17, 46]. Chen et al. [10] presents an algorithm to solve constrained continuous gate-sizing problems by Lagrangian relaxation. With convex delay/area/power models, the global continuous optimal solution is guaranteed [10, 13, 11, 36, 19]. It has been shown that with proper reformulations, LR can also handle certain statistical timing model [47] and sequential circuit [11, 48].

It's not long before researchers start working on the discrete sizing problems [24, 49, 34, 38] and one commonly used approach is called Nearest Rounding [47, 48], which obtains an optimal continuous solution first, and then rounds the size of each gate to its nearest available discrete size. However, the results of NR usually introduce large increment of delay and leave considerable room for further improvement [24].

Circuit performances are evaluated by a few factors, such as circuit delay, area, power consumption, etc. Among them, circuit delay is of vital importance in evaluating the circuit performance, since smaller circuit delay will lead to faster circuit speed. So we would like to work on the algorithms that can achieve minimum circuit delay, while previous researchers are more focused on the area or power minimization rather than delay minimization. For an example, Hu et al. [24] proposed a discrete gate sizing algorithm for area optimization under the constraints of maximum delay bounds. It uses a dynamic programming approach and reports good results. However, it is very difficult to apply their search space pruning technique to solve the problem of delay minimization. Then in [49], linear programming and some heuristics are used to solve the area optimization problem under double-sided delay constraints, but it does not address the problem of delay minimization. Livramento et al. [50] presents a dynamic programming algorithm for leakage power minimization based on Lagrangian Relaxation.

Although the problem of circuit delay minimization has been studied before, the problem is still quite challenging and there is great room to improve the existing academic algorithms. Coudert [34] presented a discrete gate sizing algorithm to minimize the maximum circuit delay by maximizing the smallest slack. It is supposed to reduce the computational cost by evaluating the gradient of subcircuits around gates whose sizes are

being considered for changes. Also power and area optimization issues are addressed. However, experimental results show the results of implementing this concept can be greatly affected by the choice of initial gate sizes, and in many cases only limited effectiveness is observed. Both [34] and [24] adopt stage-based or subcircuit-based methodologies to locally evaluate the circuit. There are other approaches, like branch-and-bound [25], dynamic programming [24, 36, 38], etc. Our observations show that for delay minimization problems, path-based delay evaluations are much more accurate and effective since all delays are path-sensitive. That is the reason why we apply the path-based delay concept in our discrete gate size selection algorithm.

In this work, by exploring such concept, we propose a fast and efficient technique for discrete gate size selection, which is able to select a proper size for each gate in a circuit from a standard cell library, with the aim to minimize the maximum delay of a circuit, and with further post-power-minimization method to keep the total power as minimum as possible.

The rest of this chapter is organized as follows. In Section 4.2, we present the problem formulation and explain the notations we used. We introduce library-cell-based selection algorithm in Section 4.3. Then in Section 4.4, we define the neighboring solution set and discrete delay difference algorithm which uses a fast local search method in measuring the delay cost. Then the post-power-optimization algorithm is presented in Section 4.5, which further reduces the power consumption by local searching. Experimental results are shown in Section 4.6. Finally a summary of this chapter is presented in Section 4.7.

4.2 The Disrete Gate Sizing Problem Formulation

In this section, we will model the circuit components and formulate the discrete gate sizing problem.

For a circuit with n sizable gates, we model it as a directed acyclic graph $G(V, E)$, where V is the set of individual gates and E is the set of wires connecting the gates. We use $v_i \in V$ to denote an individual gate, where i is the index of gate, $1 \leq i \leq n$. Then PI denotes the set of primary input and PO represents the set of primary output. Let S_i be the set of all the cells of the same gate type (like INV, NAND2, NOR3, etc) as v_i. Thus, each gate v_i has a total number of $|S_i|$ implementation options. For convenience, we use $x_i \in S_i$ to denote that the cell corresponding to size x_i is in S_i. And we define *solution point* $X = (x_1, x_2, \dots, x_n)$ as an n-tuple vector representing the sizes of all sizable gates.

We state the discrete library-cell-based sizing problem with delay and power optimization as follows: given a circuit, find an optimal solution $X^* = (x_1^*, x_2^*, \dots, x_n^*)$ such that the worst circuit delay or the maximum arrival time of the output a_o is minimized and the following constraints are satisfied:

1) The input/output transition time or slew of each gate should be within a reasonable range

2) The output load capacitance of each gate should be no more than the maximum load it is able to drive

3) To reduce the total power cost of the circuit, the leakage power of each gate should be kept as small as possible while not violating the maximum circuit delay constraint. Hence, the minimizing maximum delay problem can be formulated as following:

$$Problem: \quad Find\ X^* = (x_1^*, x_2^*, \dots, x_n^*)\ that\ Minimizes\ a_o$$

$$subject\ to: \quad a_j \le a_o, \forall v_j \in input(v_i), \forall v_i \in PO$$

$$a_j + D_i \le a_i, \forall v_j \in input(v_i), \forall v_i \in V$$

$$D_i \le a_i, \forall v_i \in PI$$

$$Min\_TT_i \le TT_i \le Max\_TT_i, \forall v_i \in V$$

$$OC_i \le Max\_output\_load\_cap_i(x_i), \forall v_i \in V$$

$$x_i \in S_i, \forall v_i \in V, i = 1,2, \dots, n$$

where a_i, D_i and TT_i refer to the arrival time, gate delay and output transition time of gate v_i, respectively. a_o conrresponds to the maximum arrival time at the output. Min_TT_i and Max_TT_i are the minimum and maximum transition time of gate v_i. OC_i is the output load capacitance of v_i, and $Max\_output\_load\_cap_i(x_i)$ is the maximum output load capacitance allowed for gate v_i with size x_i.

We refer to the lookup tables from the standard cell library to gather all the gate information, including the cell size x_i, cell capacitance c_i, max_output_load_cap, fall/rise_input_transition_time $Input\_TT_i$, fall/rise delay d_i. Linear interpolation is performed for value lookup. Mathematically, we use function f_d and f_{TT} to represent the delay and transition time function, and since they are the functions of input transition time,

output load capacitance and gate cell itself, we have: $d_i = f_{d_i}(x_i, Input\_TT_i, OC_i)$ and $TT_i = f_{TT_i}(x_i, Input\_TT_i, OC_i)$, and $Input\_TT_i$ is the input transition time of gate v_i, it is defined as the maximum transition time of all its input gates, as $Input\_TT_i = max(_{v_j \in input(v_i)}^{TT_j})$.

4.3 Optimization Methodology

The discrete gate sizing problem is NP-hard problem [26], which is slow and inefficient. So instead of searching all library cells for all the gates in the circuit, we reduce our search space to qualified cells and gates only.

First, our algorithm is a critical-path-based approach, since the critical path delay determines the worst delay of the circuit. So we put our discretization efforts on the gates on the critical path only. In addition, we only consider changing their sizes with neighboring cells, which will greatly reduce the search space and improve the efficiency of the algorithm.

Our approach begins with assigning each gate with an initial cell from the standard cell library. We allow any initial sizes as long as they do not violate the timing constraints. We may adjust the related cells in case of violation. Then for each gate along the critical path, we try different available cell sizes to see its effect on delay and determine the best one to replace, which will be discussed in the next section. And we keep exploring the gates on the same critical path. Once one critical path is fully explored with no further improvement, we will review the whole circuit and work on another critical path. With the best solution of the minimized circuit delay, we then perform the post power operation to further decrease its power consumption, which will be discussed in Section 4.5, and we apply the same local

difference method to do the fast replacement of cells that consume less power. Finally, the discrete selection algorithm will return a solution with minimum circuit delay and power consumption.

Figure 4. 1 presents the outline of the algorithm.

Algorithm 1: Library-Cell-Based Gate Selection Algorithm

Input: Circuit and Library Cell Information

Output: Solution X^* which minimizes the circuit delay and optimizes power

1. Assign each gate with an initial cell from standard cell library
2. Adjust sizes with input transition time and output load capacitance constraints
3. **Repeat**
4. Update circuit timing information
5. Update set of critical path gates
6. Call Algorithm 2 to minimize delay on critical path
7. **Until** no further improvement or stopping criterion is met
8. Record solution X as the best solution of all iterations
9. Call Algorithm 3 to do post power optimization of the X
10. Return post power optimization result X^*

Figure 4. 1. Pseudo code for Library-Cell-Based Gate Selection algorithm

4.4 Delay Minimization with Local Discrete Delay Difference

In this section, we are to describe the technique of selecting the cells of the gates along the critical path based on the calculated local discrete delay difference. First we need to define some basic terms.

A. Neighboring Solution Set

In order to narrow down the search space, instead of investigating all the standard cells for each gate, we only process the cells that are either one size smaller or one size larger than the current cell size x_i . We use x_l and x_h to represent the two cells, as $x_l \leq x_i \leq x_h$. If x_i = min cell size, $x_l = x_i$. If x_i = max cell size, $x_h = x_i$.

In discrete space, the solutions are combinatorial. Even with the narrowed-down solution space, it is still considered exponential time for testing cases with all the solutions. In order to make our optimization process within linear time and not to be trapped by the local optima, we will only change exactly one gate cell at a time in this work. If we change the size of gate v_i from x_i to x_l or x_h, and keep all the other gates unchanged, we call it a *resize* of gate v_i. Hence, we define all the solution points that could be at after any single resize at solution point X as its *neighboring solution*. Therefore, for each solution X, the set of all the neighboring solutions of X is defined as vector: $N(X) = \{W | W \; and \; X \; differ \; in \; exactly \; one \; component\}$.

B. Local Discrete Delay Difference

Then starting with current solution X with critical path delay $D(X)$, among all the neighboring vectors $N(X)$ with new delays $D(N(X))$, our problem is to find the one that maximizes $\Delta D = D(X) - D(N(X))$, which is defined as *Discrete Delay Difference*. We call the maximum delay difference as the delay difference at solution point X, denoted as $\Delta D_{max}(X)$.

To find the $N(X)$ with $\Delta D_{max}(X)$, we have to resize every gate along the critical path respectively and calcualte ΔD, which obviously requires high computational cost. So to

reduce the computational complexity, we introduce a target-path-based method to obtain the discrete delay difference, which only considers the gates directly connected to the gate being resized at the time. Since the method only evaluates the gate delay locally rather than globally, we call the obtained delay difference as *local discrete delay difference*, which will be further discussed in part D of delay cost measurement.

C. Target Local Gate Set

Then we followed the *Target Local Gate Set* defined in Chapter 2, which is greatly helpful in determining the local discrete delay difference. When considering measuring the resizing effect on a certain gate v_i, we only take two kinds of gates into consideration. First kind of gates are those gates that directly drive the gate v_i, which we call as *predecessor gates*. And second kind of gate is the one which is driven by v_i and also on the critical path, which we call it as *direct successor gate*.

D. Delay Cost Measurement

With standard cell library, the delay cost measure will need some minor changes from the one in Chapter 2. Consider the gate v_i on the critical path in Figure 2. 3, assume its current cell size is x_i , gate capacitance is c_i and arrival time is a_i, and we will see how to calculate the delay cost measurement of replacing it with a new cell x_i' with gate capacitance c_i' .

If we replace it with new cell x_i', obviously it will change the capacitance, delay, transition time and arrival time of itself, so it will change the delay, transition time and arrival time of its predecessor gates. And it will further affect its direct successor gate since

the changed transition time will affect its delay and arrival time, too. So we would like to discuss the details in the following equations:

1. For all gates $v_j \; where \; v_j \in input(v_i)$, their new Transition Time TT_j', delay d_j' and arrival time a_j' are:

$$TT_j' = f_{TT_j}(x_j, max(\underset{v_m \in input(v_j)}{TT_m}), OC_j + c_i' - c_i)$$

$$d_j' = f_{d_j}(x_j, max(\underset{v_m \in input(v_j)}{TT_m}), OC_j + c_i' - c_i)$$

$$a_j' = a_j + d_j' - d_j$$

2. For gate v_i itself, its new delay d_i' and arrival time a_i' are:

$$d_i' = f_{d_i}(x_i', max(\underset{v_j \in input(v_i)}{TT_j'}), OC_i)$$

$$a_i' = max(\underset{v_j \in input(v_i)}{a_j'}) + d_i'$$

$$TT_i' = f_{TT_i}(x_i', max(\underset{v_j \in input(v_i)}{TT_i}), OC_i)$$

3. Then for the director successor gate v_k, its new delay d_k' and arrival time a_k' are:

$$d_k' = \begin{cases} f_{d_k}(x_k, TT_i', C_k), & v_k \notin PO \\ 0, & v_k \in PO \end{cases}$$

$$a_k' = \begin{cases} a_i' + d_k', & v_k \notin PO \\ a_i', & v_k \in PO \end{cases}$$

Then the cost measurement of local discrete delay difference $\Delta D(X) = \underset{v_k \in output(v_i)}{a_k - a_k'}$.

With the cost measure explained, we listed the pseudo code of discrete delay difference algorithm in Figure 4. 2. We iteratively exam the gates along the critical path, and for each

Algorithm 2: Local Discrete Delay Difference Algorithm

Input: Circuit with solution set X with critical path CP

Output: Solution X'which minimizes the delay on CP

1. Start with initial solution set X
2. **Repeat**
3. Foreach gate along the critical Path, define the set of $N(X)$
4. Obtain Local Discrete Delay Difference $\Delta D(N(X))$
5. Find the $N(X)$ with $\Delta D_{max}(N(X))$
6. If $\Delta D_{max}(N(X)) > 0,\ X \leftarrow N(X)$
7. Update circuit timing information
8. **Until** $\Delta D_{max}(N(X)) \leq 0$
9. Return X' with minimum delay

Figure 4. 2. Pseudo code for Local Discrete Delay Difference Algorithm

gate, we explore the local discrete delay difference of each neighboring cell, and find the one which will decrease the delay most and replace the gate with the new cell. And we keep doing the process until new further improvement. We will show in our experiment that our method is able to climb out of local minima and converge to the best solution.

4.5 Post Power Optimization

After discrete selection algorithm finds a solution of the minimal circuit delay, there is still room to further reduce the total power consumption of the circuit, by possibly replacing

the gates with lower leakage power while remaining the minimum circuit delay. So we obtain a method called *Post Power Optimization* to achieve that.

However, the replacing and measuring process would be computationally extravagant if we measure the delay globally. So to improve the time efficiency, we need to reduce the resizable gate set and then follow the concept of local discrete delay difference in Section 4.4.B to minimize the total power consumption. We will start with the definitions of the resizable gate set, available cell set and delay cost measurement.

A. Resizable Gate and Available Cell Set

For the gate v_i with size x_i and leakage power p_i, we consider all the cells whose leakage power are less than p_i. And among those cells, we eliminate those that will violate the constraints of input transition time and output capacitance. So we define set of all the available cells as

$$P(X) = \{p_i' < p_i \; and \; x_i' \; does \; not \; violate \; input \; transition \; time$$

$$and \; output \; load \; capacitance \; constraint | \; \forall v_i \in \; G, i = 1,2, \dots, n\}.$$

B. Delay Cost Measurement

Again, we follow the analysis in Section 4.4.B to evaluate the delay cost measurement, but since all the resizable cells $p_i' < p_i$. And for cells with same threshold voltage, cells with less leakage power have less intrinsic capacitance, so we have their gate capacitance as $c_i' < c_i$, we can elaborate more features from those equations:

1. For all gates $v_j where \; v_j \in input(v_i)$, their new Transition Time TT_j', delay d_j' and arrival time a_j' are:

$$TT_j' = f_{TT_i}\{x_j, max(_{v_m \in input(v_j)}^{TT_m}), OC_j + c_i' - c_i\} < TT_j$$

$$d_j' = f_{d_j}\{x_j, max(_{v_m \in input(v_j)}^{TT_m}), OC_j + c_i' - c_i\} < d_j$$

$a_j' = a_j + d_j' - d_j \Rightarrow \Delta d_j = a_j' - a_j < 0$, so the arrival time of all the predecessor gates will be decreased, and will not affect the arrival time of those gates.

2. For gate v_i itself, its new delay d_i' and arrival time a_i' are:

$$d_i' = f_{d_i}'\{x_i', max(_{v_j \in input(v_i)}^{TT_j'}), OC_i\}$$

$$a_i' = max(_{v_j \in input(v_i)}^{a_j'}) + d_i'$$

Now let's take a look at $\Delta d_i = a_i' - a_i$. Concerning the value of Δd_i, we have the following three cases:

Case 1: $\Delta d_i = a_i' - a_i \leq 0$, which means resizing the gate with a smaller cell will not increase the delay along the path, so we can definitely replace it.

Case 2: $\Delta d_i = a_i' - a_i > 0$ and gate v_i is on the critical path, apparently replacing it will increase the circuit delay, so we can't replace it.

Case 3: $\Delta d_i = a_i' - a_i > 0$ but v_i is not on the critical path, then we need to investigate the effect of the increased delay to the output gates of v_i. So for each gate $v_k, where\ v_k \in output(v_i)$, we simply calculate its new added delay $\Delta d_k = a_i + \Delta d_i - a_{CI(v_k)}, where\ a_{CI(v_k)}$ is the arrival time of critical input of v_k, as critical

input is the input gate which has the largest arrival time. Then we can propagate the increased delay and continue the investigation.

The pseudo code for the three cases is defined in the function *Replaceable* in Figure 4. 3. And the post power optimization algorithm is outlined in Figure 4. 4.

Function Replaceable (gate v_i, delay differenc Δd_i)

Input: gate v_i and its gate delay difference Δd_i

Output: *True* if the circuit delay will not be increased with Δd_i. *False* if else.

```
1.  If (Δd_i ≤ 0)
2.      Return true.
3.  Else
4.      If (v_i is on Critical Path)
5.          Return false.
6.      Else
7.          Foreach gate v_k, where v_k ∈ output(v_i)
8.              Δd_k = a_i + Δd_i − a_CI(v_k)
9.              If (Replacable(v_k, Δd_k) == false)
10.                 return false.
11.         Return true.
```

Figure 4. 3. Pseudo code for Function *Replacable*

Algorithm 3: Post Power Optimization Algorithm

Input: Solution X with minimum circuit delay

Output: Solution X^* with same minimal circuit delay but optimized power

```
1.  Start with initial solution set X
2.  Repeat
3.      Define the set of P(X)
4.      Foreach gate v_i
5.          Foreach cell x_i' in P(X), start from minimum power consumption
6.              If replaceing x_i' will not violate the constraints
7.                  Obtain Δd_i(P(X))
8.                  If(Replacable ( v_i, ΔD_max(P(X)))== true)
9.                      X ← P(X), break
10.             Else
11.                 Continue
12.     Update circuit timing information
13. Stop until no further improvement or stop requirement is met
14. Return X* with minimum circuit delay and optimized power
```

Figure 4. 4. Pseudo code for Post Power Optimization algorithm

4.6 Experimental Results

To evaluate the proposed discrete gate sizing technique, we implemented the algorithm in C++ and the experiments are performed on a 3.4 GHz Intel Core i-7 computer. We use the standard cell library provided by the ISPD 2012 Discrete Gate Sizing Contest [6]. The ISPD 2012 standard cell library includes 11 combinational gates, and each type of gates has 3 threshold voltage (Vt) levels and 10 sizes (drive strengths) for each Vt level. In our

experiment, for better comparison, we only consider the gates at the same Vt level, limiting each gate with 10 choices of gate size, whose cell ID range from 0 to 9, as 0 is smallest cell size and 9 is the largest cell size. Then we use ISCAS ’85 benchmark circuits, which are commonly used for circuit analysis and optimization (see [51, 52]). To judge the efficacy of our discrete gate sizing algorithm, we consider the algorithm performance from three different aspects, which are delay, power, and runtime costs.

First on concerning the delay performance, for each circuit, we did 6 test cases with different starting gate sizes. We set all the gates with the initial cell ID = 0, 2, 4, 6, 8 and 9, then compare the final circuit delay and power after running the algorithm. In Table 4. 1, we listed the original circuit delay with different initial cell sizes, and the minimum circuit delay after running our algorithm. We can see that for each circuit, the final circuit delay ranges from 1.3% to 7.1%, with an average of 4.4%, which means that the algorithm is able to reach a final solution whose circuit delay is very close to each other, no matter what the initial cell sizes are.

Then in Table 4. 2, we listed the original circuit power with different initial cell sizes and the circuit power after running the algorithm. And it shows that, the final circuit power ranges from 0.6% to 10%, with an average of 4.2%, which also proves that the algorithm is able to reach a final solution whose circuit power is very close to each other, no matter what the initial cell sizes are. So concerning both the result of circuit delay and power, we can conclude that our algorithm is able to reach a final solution which is very close to the optimal global solution, no matter what the initial sizes are.

Table 4. 1. Minimum Circuit Delay Comparison with Different Initial Cell Size.
Delays are in *ps*.

Circuit	Initial Cell ID = 0		Initial Cell ID = 2		Initial Cell ID = 4		Initial Cell ID = 6		Initial Cell ID = 8		Initial Cell ID = 9		Max After Delay	Min After Delay	Max/ Min -1
	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After			
C432	1012	701	919	701	861	703	1090	694	1328	698	1371	699	703	694	1.3%
C499	682	557	672	531	639	558	905	558	973	569	1001	538	569	531	7.2%
C880	1495	842	1037	858	1051	857	1360	840	1507	845	1577	844	859	840	2.3%
C1355	1089	959	1024	981	1062	936	1374	975	1521	984	1638	977	981	936	4.8%
C1908	1524	1167	1194	1018	1187	1067	1353	1117	1526	1098	1574	1098	1066	1018	4.7%
C2670	1667	1015	1183	1009	1104	994	1279	1016	1516	1019	1571	1027	1027	994	3.3%
C3540	2547	1477	1780	1467	1803	1489	1931	1506	2078	1451	2216	1461	1506	1451	3.8%
C5315	2103	1320	1478	1342	1426	1354	1615	1330	1874	1316	2012	1366	1366	1316	3.8%
C6288	5723	4089	5636	4086	4306	4179	4605	4096	5642	4279	6210	4320	4320	4086	5.7%
C7552	1710	1266	1316	1250	1267	1182	1501	1240	1643	1221	1724	1200	1266	1182	7.1%
Avg.															4.4%

Table 4. 2. Final Power Comparison with Different Initial Cell Size.

Circuit	Initial Cell ID = 0		Initial Cell ID = 2		Initial Cell ID = 4		Initial Cell ID = 6		Initial Cell ID = 8		Initial Cell ID = 9		Max After Delay	Min After Delay	Max/ Min -1
	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After			
C432	1696	2248	3748	2248	6908	2280	18068	2280	53780	2372	88460	2364	2372	2248	5.5%
C499	3664	5072	7232	5100	14816	4864	33696	4952	74720	4896	107232	4848	5072	4848	4.6%
C880	3028	3352	7212	3288	13408	3264	33308	3364	87936	3328	136564	3592	3592	3264	10.0%
C1355	4464	5068	10688	4972	19766	5312	47424	5220	143680	5000	256416	5336	5336	4972	7.3%
C1908	6116	6448	14820	6824	28228	4288	71524	6468	265544	6652	490556	6656	6824	6448	5.8%
C2670	8668	9136	20768	9188	39420	9352	104568	9152	380312	9204	668100	9220	9352	9136	2.4%
C3540	11844	12872	29000	12944	53520	12980	132180	13040	409384	13056	682564	13032	13056	12872	1.4%
C5315	19608	20628	43040	20548	80896	20540	202280	20548	641764	20572	1010088	20512	20628	20512	0.6%
C6288	19588	23432	48064	23520	84288	22952	213248	23436	681440	23080	1059376	22936	23520	22936	2.5%
C7552	25424	26160	62048	26248	117056	26600	304068	26304	1157484	26364	2187764	26428	26600	26160	1.7%
Avg.															4.2%

Table 4. 3. Delay and Power Comparison between Our Algorithm (NEW) and Liu's Algorithm [36]. Delays are in *ps.*

Circuit	# of gates	New		Liu [36]		New vs. Liu [36]	
		Delay	Power	Delay	Power	Delay	Power
c432	160	694	2280	870.4	2684	-20.27%	-15.05%
c499	202	531	5100	627.2	3664	-15.34%	39.19%
c880	383	840	3364	1052.7	4574	-20.21%	-26.45%
c1355	546	936	5312	960.5	5176	-2.55%	2.63%
c1908	880	1018	6824	1074.4	6246	-5.25%	9.25%
c2670	1193	994	9352	1068.3	9388	-6.95%	-0.38%
c3540	1669	1451	13056	1687.9	17618	-14.04%	-25.89%
c5315	2307	1316	20572	1338.9	23904	-1.71%	-13.94%
c6288	2416	4086	23520	3837.4	30484	6.48%	-22.84%
c7552	3512	1182	26600	1227.9	27450	-3.74%	-3.10%
Avg.	1327	1304.8	11598	1374.56	13118.8	-8.36%	-5.66%

In Table 4. 3, we listed the minimum delay and corresponding power from our algorithm in Table 4. 1, and compared it with the previous work from Liu [36]. The work of [36] employs bi-directional solution search which searches for potential solution in the backward direction and prune the solution in the forward search. However, with the new standard cell library where the change of the cell size may affect its input transition time, one problem with this method is that it is not able to predict its effect on its input side information, so it may cause great imprecision in searching and pruning solution. We implemented the JRR from [36] to obtain the minimum delay solution first and then minimize power with the minimum delay constraint. As the data in Table 4. 3, we can see the delay and power of our algorithm is 8.36% and 5.66% less than that of [36].

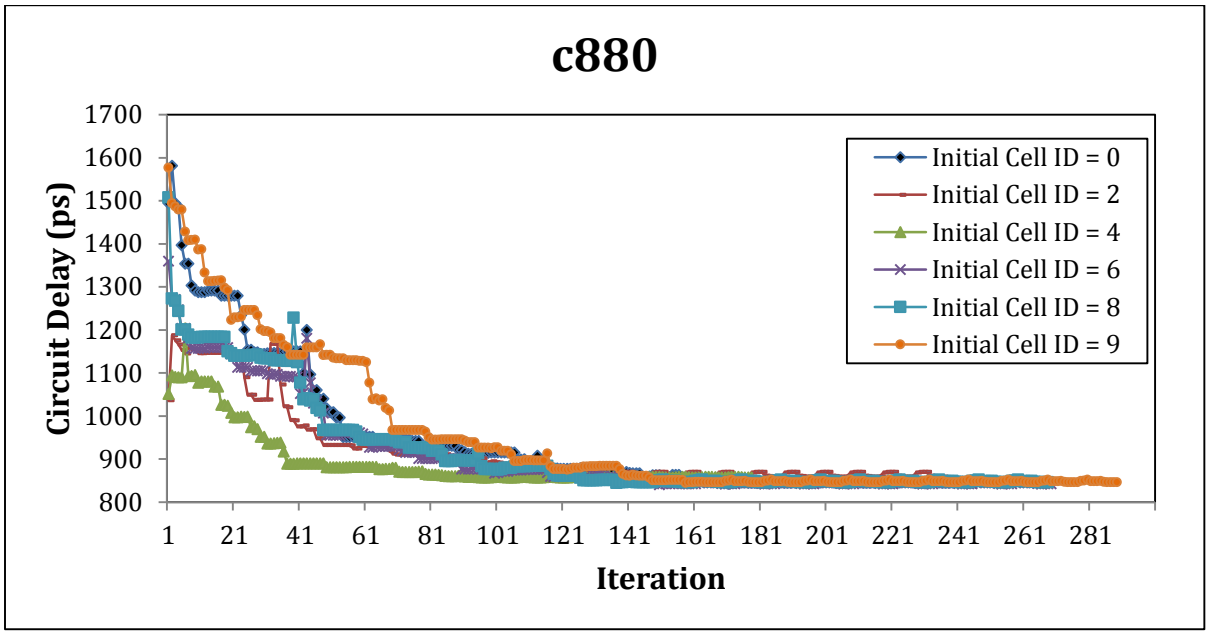

Figure 4. 5. Delay Sequence of Circuit c880, with Initial Cell ID = 0, 2, 4, 6, 8,9

Figure 4. 5 shows the delay sequence of circuit c880, with different initial cell IDs. As we can see from the figure, with different initial circuit delays, it will converge to a close final circuit delay as the iteration goes on, and it takes fewer iterations to reach the final solution if the initial cell sizes are closer to the optimal solution.

Figure 4. 6 and Figure 4. 7 show the delay sequence of circuit c880 with initial cell ID = 0 and 9, and for better illustration, we only mark the delay with the first 100 and 50 iterations. It's shown that at some iterations, the circuit delay may be allowed to increase temporarily to make a better chance to lead to the better solution finally. Similar cases are found in sequences of other test cases.

To evaluate the runtime performances, Table 4. 4 shows the runtime of each circuit with different initial cell sizes. Figure 4. 8 shows the runtime versus the # of gates. And the linear curve is plotted for comparison.

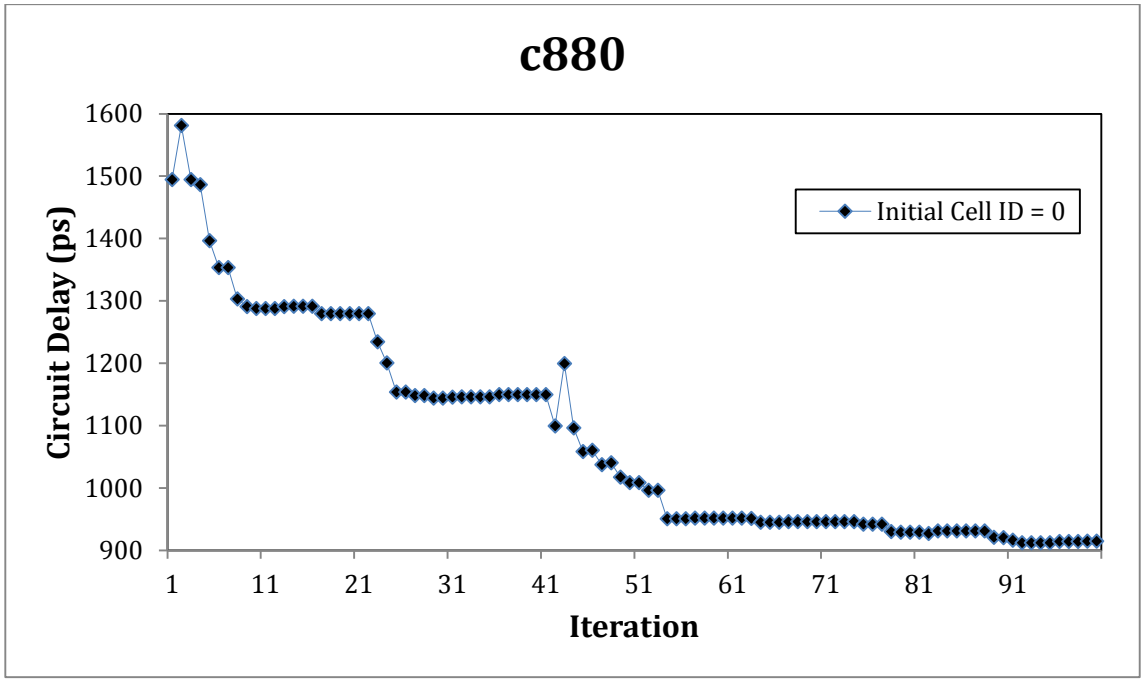

Figure 4. 6. Delay Sequence of Circuit c880, with Initial Cell ID =0

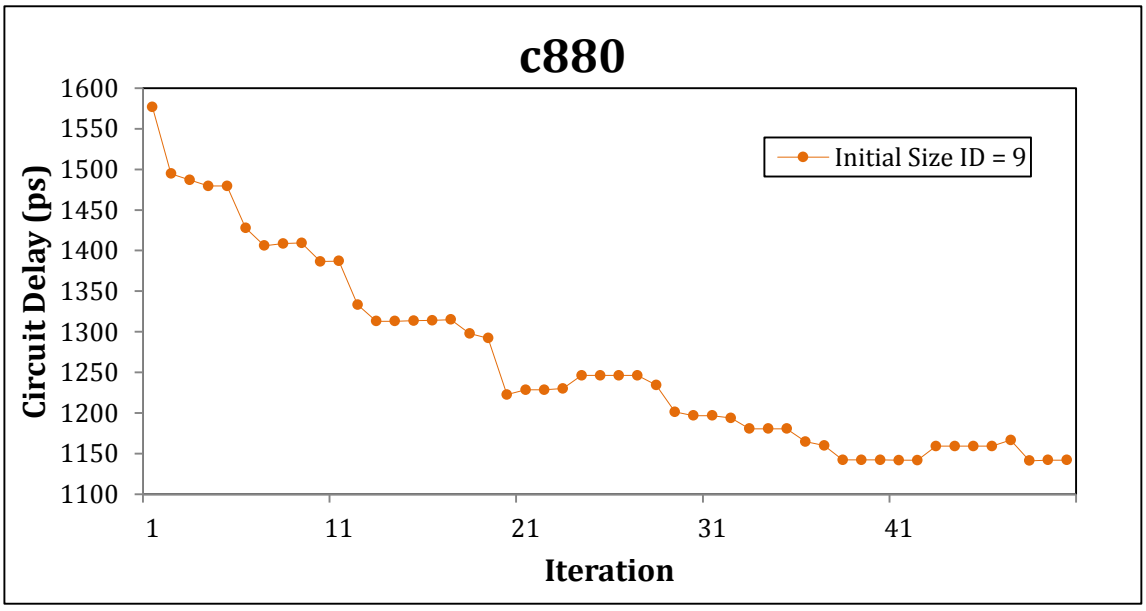

Figure 4. 7. Delay Sequence of Circuit c880, with Initial Cell ID =9

Table 4. 4. Runtime Comparison between Different Circuits with Different Initial Cell Size. *Runtimes* are in *seconds (s).*

Circuit	#of gates	Initial Cell ID = 0	Initial Cell ID = 2	Initial Cell ID = 4	Initial Cell ID = 6	Initial Cell ID = 8	Initial Cell ID = 9	Avg.
c432	160	0.04	0.03	0.04	0.05	0.04	0.05	0.04
c499	202	0.01	0.00	0.01	0.02	0.02	0.01	0.01
c880	383	0.05	0.04	0.04	0.04	0.05	0.06	0.04
c1355	546	0.03	0.06	0.06	0.06	0.06	0.08	0.06
c1908	880	0.11	0.15	0.11	0.10	0.15	0.16	0.13
c2670	1193	0.16	0.16	0.15	0.16	0.18	0.18	0.16
c3540	1669	0.41	0.14	0.21	0.44	0.44	0.45	0.35
c5315	2307	0.50	0.48	0.51	0.27	0.55	0.48	0.47
c6288	2416	2.16	2.72	2.36	2.32	2.17	2.22	2.32
c7522	3512	0.77	0.59	0.68	0.75	0.84	0.88	0.75
Avg.	1326.8	0.4241	0.4365	0.4156	0.4209	0.4506	0.4545	0.4337

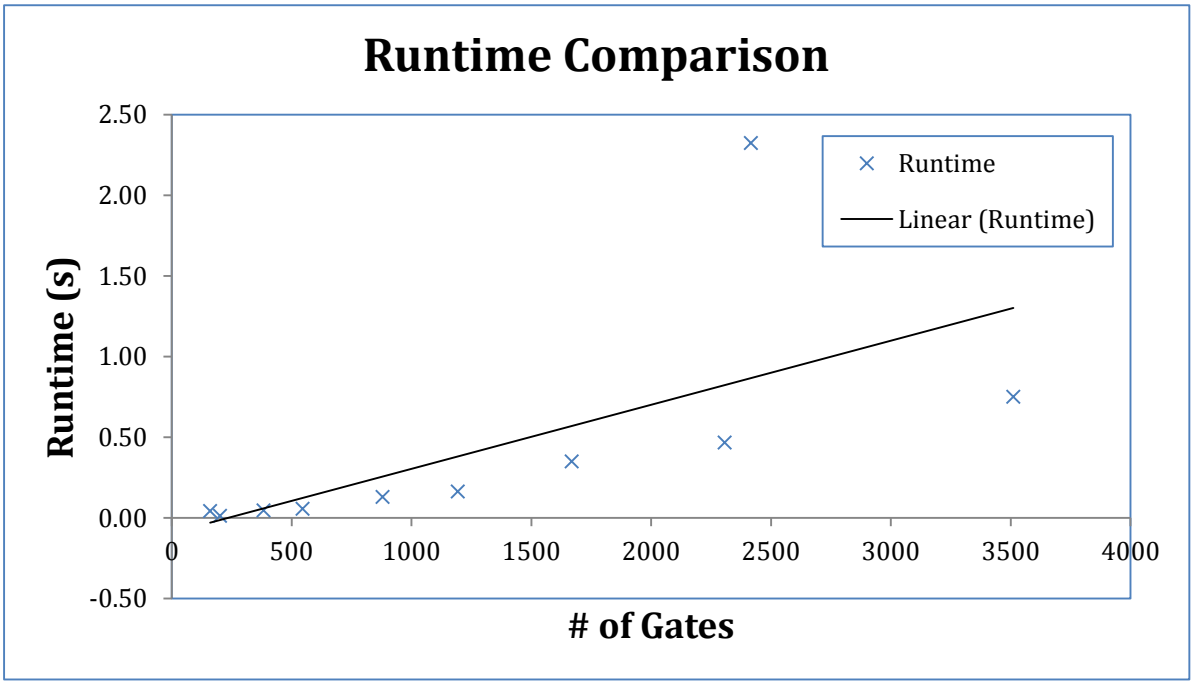

Figure 4. 8. The Comparison between Runtime and # of Gates

Table 4. 5. Runtime Comparison between Our Algorithm (New) with Liu's Algorithm [36].

Runtimes are *seconds(s).*

Circuit	#of gates	New	Liu [36]	New vs. Liu [36]
c432	160	0.04	0.02	206.67%
c499	202	0.01	0.063	19.05%
c880	383	0.04	0.059	75.71%
c1355	546	0.06	0.96	5.80%
c1908	880	0.13	0.128	101.17%
c2670	1193	0.16	0.099	165.15%
c3540	1669	0.35	2.177	16.00%
c5315	2307	0.47	1.063	43.92%
c6288	2416	2.32	0.434	535.60%
c7522	3512	0.75	1.27	59.11%
Avg.	1326.8	0.4337	0.6273	69.14%

Table 4. 5 shows the runtime comparison between our algorithm (New) with Liu's algorithm [36]. On average, our algorithm saves 30.84% time compared with Liu's algorithm.

4.7 Summary

In this work, we have presented a library-cell-based discrete gate sizing selection method for delay minimization and power optimization afterwards. Our algorithm is able to work fast and efficiently in minimizing the circuit delay by evaluating the local discrete delay difference of the target local gate set, which greatly reduces the search space. Then we further minimize the power consumption while retaining the minimized circuit delay using the similar local search on cell replacement.

We implement our algorithm on ISCAS ’85 benchmarks with ISPD 2012 standard cell library, and the experimental results show that this approach is able to converge to a near-optimal solution with different initial sizes. In addition it is able to handle about 4000 gates within a second, and runtime is almost linear to the number of gates.

Chapter 5

Discrete Gate Sizing from Standard Library Cell with Modified Elmore Delay Model and Lagrangian Relaxation

In this chapter, we present how to modify the classic Elmore delay model to characterize the lookup tables from standard cell library, and revise the existing continuous Lagrangian relaxation method to solve the discrete delay and power optimization problem. Elmore delay model was thought to be an outdated model because of its inaccuracy and its inability to take into account input waveforms. However, with our revision, it is able to adapt to the features of the updated lookup tables from the standard cell library, and keep its simple elegance at the same time. In our proposed method, we use a two-phase method that first finds the optimal continuous solution that minimizes the circuit delay by using the Lagrangian relaxation approach, which is fast in convergence. And then we project the continuous solution to discrete solution using nearest rounding with refinement and further reduce the power consumption by using the post power optimization method. In our experiment, compared with the heuristic we proposed in Chapter 4, the delay of our new method is within 3.8% or more and the power is averagely 0.1% less, which proves that the modified Elmore delay model is effective in evaluating circuit performance.

5.1 Introduction

With the development of vary large scale integrated (VLSI) technology, it has been more and more important to have the circuit designs with optimized performance. So people having been researching for the techniques to optimize the circuit for a long time. There are a few gate sizing techniques; however, they all have certain limitations. Some techniques can only handle continuous solutions, while others focus on optimizing power/area rather than optimized the circuit delay directly. And in this work, we are focusing on the problem of discrete gate sizing with delay minimization. Besides, we will modify the classic Elmore delay model, which is replaced by lookup tables now because of its inefficiency and inaccuracy, and we will solve the discrete gate sizing problem using Lagrangian relaxation with the new model. The problem of discrete gate sizing problems is to assign a proper cell to each gate from a few options in the standard cell library. We intend to solve the constraint free delay minimization problem by assigning discrete gate cells from the standard cell library, which few people have done before.

First, let us have a quick review of the gate sizing problem, which has been extensively studied. A lot of former researchers were focusing on solving continuous sizing problems, which is unrealistic in real life and impractical to manufacture. Fishburn et al. [8] used convex optimization technique TILOS to size transistors. Chen et al. [10] solved constrained continuous gate and wire sizing problems using Lagrangian relaxation, which was later refined by Wang et al. in [11]. Tennakoon et al. [13] utilized a gradient based approach with Elmore delay model.

However, from [24], discretization of continuous solutions may lead to timing violations for sparse cell libraries. So nowadays people are trying to solve sizing problems

directly, which is more practical and accurate. But with the complexity of the problem, only until recently, researchers started working on the discrete sizing problems. Coudert [34] proposed a randomization-based greedy search algorithm. Hu et al. [24] applied a dynamic programming approach guided by continuous solution.

Second, when concerning the models in solving gate sizing problems, Elmore delay model [2] has significant impact in the history. Elmore delay is a simply approximation in calculating circuit delay through an RC network. The delay of each gate is its resistance times its download capacitance. Because of its simplicity, Elmore delay model has been widely used to evaluate circuit performance and solving sizing problems. Chen et al. [10] built gate and wire delay more based on Elmore model. And in [13], Elmore delay model is adopted to formulate the problems.

But with modern complex timing models, classic Elmore delay cannot fully characterize the features of current standard cell libraries. Thus nowadays, researchers tend to use lookup tables for describing gate delays, transition times, etc. But it is hard for researchers to model the lookup table. Ozdal et al. [53] proposed a graph model to capture library cell feature. However, we find that the Elmore delay model is still valuable and effective for solving gate sizing problems with its simplicity, if we properly modify the model to fit the features of the lookup tables, which will be discussed in the next sections.

Furthermore, circuit performance is evaluated by delay, area, power, etc. Among them, circuit delay is of vital importance, since smaller circuit delay will lead to faster circuit speed. However, previous researchers primarily focus on optimizing power or area rather than delay. Nguyen et al. [54] used linear programming to minimize dynamic and static power. Livramento et al. [39] used a state-of-the-art method to minimize leakage power

with delay constraint. Liu et al. [36] proposed a systematic combinatorial approach with threshold voltage assignment.

Constraint free delay minimization, which is addressed in this chapter, is very important but rarely studied in literature; only few researchers [10, 34, 40] have perform work on this, but they are either limited to continuous sizing [10], outdated delay model [34], or much more complicated approach than our proposed one [40].

In this work, we have proved that the classic Elmore delay model can be modified to characterize the feature of the discrete lookup table, and furthermore, with its simplicity, we present an algorithm that is able to solve the gate sizing problems using Lagrangian relaxation with the modified Elmore delay model, which is efficient and guaranteed to converge to optimal global solution.

The rest of this chapter is organized as follows. In Section 5.2, we first define the notations and terminology that we use in this work. And in Section 5.3 we show how to modify the classic Elmore Delay model to fit the lookup tables from ISPD 2012 standard cell library. Then in Section 5.4, we present our algorithm to solve the delay minimization problem using Lagrangian relaxation with the modified Elmore delay model. In Section 5.5, we project the continuous solution to discrete field using nearest rounding with delay cost refinement, and we apply the post power optimization later to further reduce power consumption while maintaining the minimum circuit delay. The experimental results are shown in Section 5.6. And the summary follows in Section 5.7.

5.2 Preliminaries

Before we present our algorithm, we first define the notations and terminology that we use in this work.

For simplicity, we only focus on combinational circuits and ignore all the latches. And we only consider gates and ignore wires since current standard libraries do not provide lookup table for wires, so we focus on the sizing of gates only.

For a circuit with n sizable gates, we model it as a directed acyclic graph (DAG). The DAG is represented as $G(V, E)$, where V is the set of individual gates and E is the set of wires connecting the gates. And we use $v_i \in V$ to represent an individual gate in the DAG, where i is the index of the gate, as $1 \leq i \leq n$. Concerning a circuit with s primary inputs,

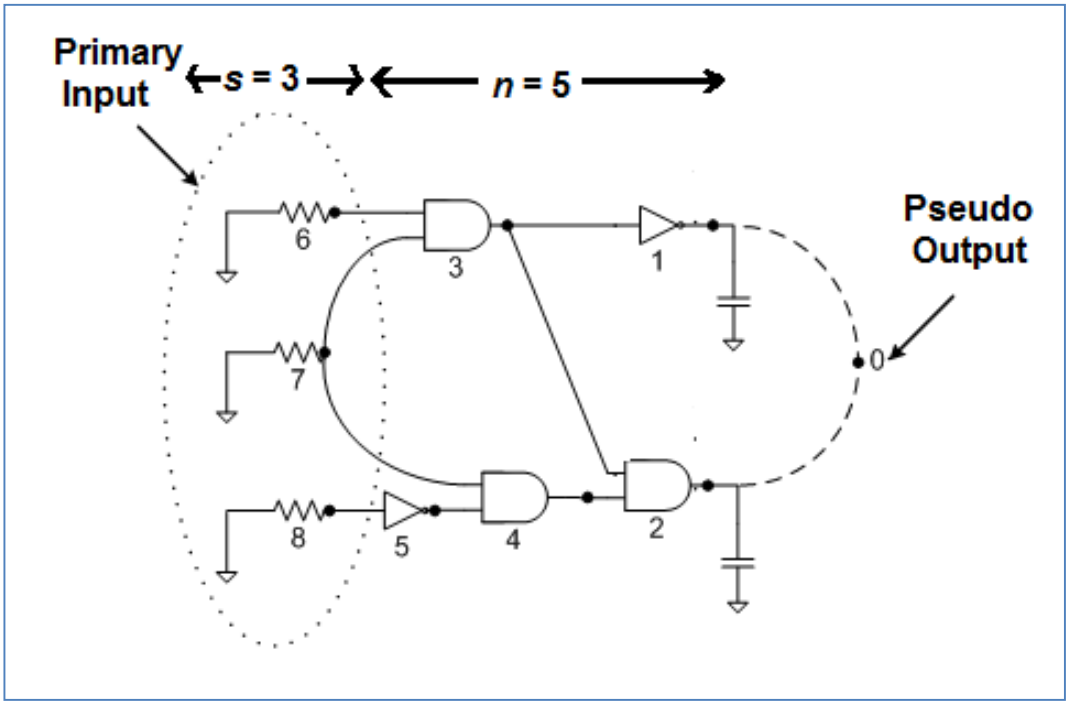

Figure 5. 1. An illustration of circuit notations

we use PI to represent the set of primary inputs. And we add a pseudo output PO to connect all the primary outputs. Then we use reverse topological sorting to label all the gates from output to input, so that if gate i is more closed to the pseudo output than gate j, then $i < j$. Thus, in the circuit with n resizable gates and s primary inputs, the PO is labeled as gate 0, all the resizable gates are labeled from $1\ to\ n$, and the primary inputs are labeled from $n + 1$ to $n + s$. As an example, Figure 5. 1 shows a circuit with 5 sizable gates and 3 inputs.

Then we use $X = (x_1, x_2, \dots, x_n)$ as an n-tuple vector to represent the discrete sizes of all sizable gates in the circuit. And each $x_i, where\ i = 1, \dots, n,$ represents a discrete size for gate v_i. x_i is referred as the i-th component of X, and X is simply referred as a $solution\ point$ in our following discussions. And for each type of gate i, it comes with a lower bound and an upper bound for its size, denoted as L_i and U_i, and we have $L_i \leq x_i \leq U_i$.

For each gate i, where $1 \leq i \leq n,$ we use $input(i)$ to represent the set of indexes of components that are directly connected to the inputs of component i. And we use $output(i)$ to represent the set of indexes of components that are directly connected to the outputs of component i. In addition, for each component i, where $1 \leq i \leq n + s,$ $a_i\ and\ D_i$ are used to denote its arrival time and delay. OC_i is donoted as its output load capacitance, and we use $Input\_TT_i$ to denote the input transition time of gate i. We use the modified Elmore delay model to calculate the component delay, which will be discussed in detail in the next section.

5.3 Modified Elmore Delay Model

In this section, we modify the classic Elmore delay model so that it fits the characters of the look up table from ISPD 2012 [6] standard cell library.

First we would like to review the timing model from ISPD 2012 standard cell library. The cell timing delay and output transition time are implemented as lookup tables, and these lookup tables take the output load capacitance and input transition time as inputs and generate gate delay and output transition time. Mathematically, we use function $d_i = f_{d_i}(x_i, Input\_TT_i, OC_i)$ and $TT_i = f_{TT_i}(x_i, Input\_TT_i, OC_i)$ to represent the delay and output transition time of gate i. And we can use a simple 2-dimensional linear interpolation model for specific values from the table.

Then let us see how to modify the classic Elmore delay to feature the ISPD 2012 library.

A. Delay and transition time

Each cell from ISPD 2012 standard cell library comes with fall/rise delay and fall/rise transition time, which are delays and transition times for fall or rise transitions separately. Even though it comes with four types of look up tables, their values at each parameter are so close that we can use only one of the four to represent all of them. On the other hand, since Elmore delay model only deals with delay, we can use the delay of a gate to approximate its output transition time. So we have the following equations for defining the output transition $\mathrm{TT_i}$ and input transition time $\mathrm{Input\_TT_i}$ using delay only:

$$TT_i = D_i \tag{5. 1}$$

$$Input\_TT_i = max(\underset{j \in input(i)}{TT_j}) = max(\underset{j \in input(i)}{D_i}) \quad (5.\ 2)$$

B. Gate model

We keep modeling the gates as resistor-capacitor (RC) only, as shown in Figure 5. 2. However, we need to make some revisions on the gate capacitance c_i and gate resistance r_i as follows:

$$c_i = \hat{c}_i \times x_i, \text{ where } \hat{c}_i = 1/(Number\ of\ gate\ inputs) \quad (5.\ 3)$$

$$r_i = \hat{r}_i / x_i + r_{i0} \quad (5.\ 4)$$

In the two equations above, $\hat{c}_i$ is the unit size capacitance, which is the reciprocal of the number of gate inputs, for examples, if gate i is an inverter, then $\hat{c}_i = 1$; if gate i is a two-input NAND gate, then $\hat{c}_i = 1/2$. And $\hat{r}_i$ is the unit output resistance, and r_{i0} is the intrinsic resistance.

C. Maximum output load capacitance limit

Each cell from the standard library comes with a feature called maximum output load capacitance, which defines the maximum output load capacitance that the output pin is able to drive.

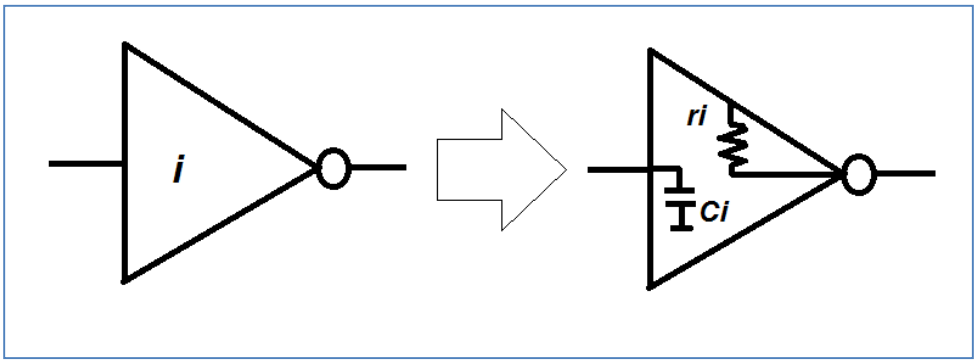

Figure 5. 2. Example of RC model of an inverter

And we find that the maximum output load capacitance is a linear function of the cell size from ISPD 2012 library. Thus, we define the maximum output capacitance for gate i as:

$$Max\_cap\_load_i = Unit\_cap_i \times x_i \tag{5. 5}$$

where $Unit\_cap_i$ is the maximum capacitance that can be drive by a unit size cell.

D. Modified Elmore Delay Model

By deep reviewing of the lookup tables, we make some minor modifications on the Elmore delay model, and the new delay for gate i, where $1 \leq i \leq n$, is:

$$\begin{aligned} D_i &= \alpha_i(Input\_TT_i) \times \beta_i(OC_i) \times (r_i \cdot OC_i + D_0) \\ &= \alpha_i(Input\_TT_i) \times \beta_i(OC_i) \times \left(\left(\frac{\hat{r}_i}{x_i} + r_{i0}\right) \cdot OC_i + D_{i0}\right) \\ &= \alpha_i \cdot \beta_i \cdot \left(\left(\frac{\hat{r}_i}{x_i} + r_{i0}\right) \cdot OC_i + D_{i0}\right) \end{aligned} \tag{5. 6}$$

In the equation above, we define D_{i0} as the intrinsic delay of gate i, and it is the delay of gate i when its output load capacitance is 0.

As we can see from the lookup table of ISPD 2012 standard cell library, with the same output load, the gate delay slightly increases if the input transition time increases. So we use $\alpha_i(Input\_TT_i)$ as a function of the input transition time of gate i to reflect this feature. We can use define $\alpha_i(Input\_TT_i)$ either as a linear function of $Input\_TT_i$ or as a piecewise linear function for more precision. Similarly, the delay may slightly change with the output load capacitance, so we use $\beta_i(OC_i)$ to reflect the change. And we use α_i and β_i to represent the two parameters $\alpha_i(Input\_TT_i)$ and $\beta_i(OC_i)$ for short.

We list the modified Elmore delay parameters in Table 5. 1. We refer to the ISPD 2012 standard cell library to gather all the information of the gate cells and formulate those parameters such as cell sizes, unit resistance, unit capacitances, and intrinsic delay, etc.

Table 5. 1. Modified Elmore Delay Parameters

Gate capacitance	$c_i = \hat{c}_i \times x_i,$ where $\hat{c}_i = 1/(Number\ of\ gate\ inputs)$
Gate resistance	$r_i = \hat{r}_i / x_i + r_{i0}$
Gate delay	$D_i = \alpha_i(Input\_TT_i) \times \beta_i(OC_i) \times (r_i \cdot OC_i + D_{i0}) = \alpha_i \cdot \beta_i \cdot \left((\hat{r}_i/x_i + r_{i0}) \cdot OC_i + D_{i0}\right)$
Transition time	$TT_i = D_i$
Input transition time	$Input\_TT_i = max\left(_{j \in input(i)}^{TT_j}\right) = max(_{j \in input(i)}^{D_i})$
Maximum output capacitance	$Max\_cap\_load_i = Unit\_cap_i \times x_i$

E. Gate Example

We use the inverter cell in01m01 in Table 1. 2 as an example to compare our new delay model with the original lookup table. We list all the new delay model parameters in Table 5. 2. Since the transition time constraint set the slew between 5ps – 300ps and the maximum output load capacitance of cell in01m01 is 14.4ff, we only consider the input transition time between 5ps – 300ps and output load capacitance less than 16ff. And the new model delay for inverter cell in01m01 is listed in Table 5. 3. Table 5. 4 shows the delay comparison between Modified Elmore delay model and the classic Elmore delay model. They are very close when the input transition time and output load capacitance are

Table 5. 2. Modified Elmore Delay Parameters for Inverter Cell in01m01

$\hat{c}_i$	$1ff$
$\hat{r}_i$	$4.465\ k\Omega$
r_{i0}	$0\ k\Omega$
D_{i0}	$10.17\ ps$
$\alpha_i(Input\_TT_i)$	$\begin{cases} 0.8 & Input\_TT_i \in [5,30) \\ 0.97 & Input\_TT_i \in [30,50) \\ 1.14 & Input\_TT_i \in [50,80) \\ 1.35 & Input\_TT_i \in [80,140) \\ 1.7 & Input\_TT_i \in [140,200) \\ 2.05 & Input\_TT_i \in [200,300) \\ 2.5 & Input\_TT_i = 300 \end{cases}$
$\beta_i(OC_i)$	$\begin{cases} 1.75 & OC_i \in [0,1) \\ 1.55 & OC_i \in [1,2) \\ 1.4 & OC_i \in [2,4) \\ 1.23 & OC_i \in [4,8) \\ 1.1 & OC_i \in [8,16) \\ 1 & OC_i = 16 \end{cases}$
$Unit\_cap_i$	$14.4\ ff$

Table 5. 3. Modified Elmore Delay for Inverter Cell in01m01

Output Load	Input Transition Time						
Capacitance	5	30	50	80	140	200	300
0	10.17	17.2636	20.2892	24.0266	30.2558	36.4849	44.4938
1	14.64	22.0112	25.8689	30.6342	38.5764	46.5186	56.73
2	19.1	25.9378	30.4836	36.099	45.458	54.817	66.85
4	28.03	33.4426	39.3037	46.5438	58.6107	70.6776	86.1923
8	45.89	48.9646	57.5461	68.1467	85.8143	103.482	126.198
16	81.61	79.1617	93.0354	110.174	138.737	167.301	204.025

Table 5. 4. Delay Comparison between Modified Elmore Delay Model and Classic Elmore Delay Model

Delay Comparison (Modified Elmore Delay Model vs. Classic Elmore Delay Model)							
Output Load	Input Transition Time						
Capacitance	5	30	50	80	140	200	300
0	0.00%	5.72%	0.34%	-3.16%	-6.62%	-5.80%	-6.19%
1	0.00%	5.57%	1.09%	-1.72%	-3.41%	-1.92%	-2.34%
2	0.00%	2.48%	0.64%	-1.93%	-2.78%	-0.33%	-0.34%
4	0.00%	-2.33%	0.24%	-0.25%	-0.51%	2.95%	4.49%
8	0.00%	-6.02%	0.83%	5.60%	8.15%	12.44%	15.46%
16	0.01%	-9.85%	0.26%	9.91%	20.47%	28.62%	32.88%

far less than the maximum value, and they may vary a bit when they are reaching the maximum limit. Since in most cases, the input transition time and output load capacitance are far below the maximum value, these variations are acceptable.

5.4 Minimizing Circuit Delay with Lagrangian Relaxation

In this section, we solve the problem of minimizing the maximum delay in continuous domain with Lagrangian relaxation. We use the modified Elmore delay subject to the transition time constraints and output load capacitance constraints.

In part A, we first represent how to formulate the problem. And in part B, C and D, the primal problem is solved by introducing Lagrangian multipliers and solving its sub and dual problems.

A. Problem Formulation

The delay of the circuit can be defined as the maximal delay of all the paths in the circuit. And minimizing the circuit delay is the same as minimizing the arrival time of the pseudo output. So the primal problem of minimizing maximum delay can be formulated as following:

$$
\begin{aligned}
&Primal\ Problem: \quad Find\ X^* = (x_1^*, x_2^*, \dots, x_n^*)\ that\ Minimizes\ a_O \\
&subject\ to: \ a_j \le a_o, \qquad v_j \in PO \ \ /* \ outputs \ */ \\
&\qquad a_j + D_i \le a_i, \forall v_j \in input(v_i), \forall v_i \in V \\
&\qquad D_i \le a_i, \forall v_i \in PI \ \ /* \ inputs \ */ \\
&\qquad L_i \le x_i \le U_i, \ \forall v_i \in V \\
&\qquad Min\_TT_i \le D_i \le Max\_TT_i, \forall v_i \in V \ \ /* \ transition\ time\ constraints \ */ \\
&\qquad OC_i \le Unit\_cap \times x_i, \forall v_i \in V \ \ /* \ output\ capacitance\ constraints \ */ \\
&\qquad \forall v_i \in V, i = 1,2, \dots, n
\end{aligned}
\tag{5. 7}
$$

B. Lagrangian Relaxation

We use Lagrangian relaxation procedure to solve the primal problem. For each constraint on arrival time, we introduce a nonnegative Lagrange multiplier λ to formulate a new subproblem L_λ as follows:

$$
L_\lambda = a_0 + \sum_{j \in PO} \lambda_{j0}(a_j - a_0) + \sum_{i=1}^{n} \sum_{j \in input(i)} \lambda_{ji}(a_j + D_i - a_i) + \sum_{i=n+1}^{n+s} \lambda_i (D_i - a_i) \tag{5. 8}
$$

Then the Lagrangian relaxation subproblem associated with the Lagrange multipliers λ is:

$$LRS/\lambda: Minimize\ L_\lambda(x,a)$$

$$Subject\ to: L_i \le x_i \le U_i, \qquad \forall v_i \in V$$

$$Min\_TT_i \le D_i \le Max\_TT_i, \forall v_i \in V$$

$$OC_i \le Unit\_cap \cdot x_i, \forall v_i \in V$$

$$\forall v_i \in V, i = 1,2,\dots,n \qquad (5.\ 9)$$

Where

$$L_\lambda(x,a) = a_0 + \sum_{j \in PO} \lambda_{j0}(a_j - a_0) + \sum_{i=1}^{n} \sum_{j \in input(i)} \lambda_{ji}(a_j + D_i - a_i) + \sum_{i=n+1}^{n+s} \lambda_i(D_i - a_i)$$

From the Kuhn-Tucker conditions [55], which imply $\partial L/\partial a_j = 0, 1 \le j \le n+s$ for optimal solutions of the primal problem, we have the following optimality conditions on λ:

$$1 = \sum_{j \in input(0)} \lambda_{j0} \qquad (5.\ 10)$$

$$\sum_{i \epsilon output(k)} \lambda_{ki} = \sum_{j \epsilon input(i)} \lambda_{ij}\,, where\ 1 \le k \le n+s \qquad (5.\ 11)$$

Then for λ satisfying the conditions, we can simplify LRS/λ to the following problem:

LRS/u: $Minimize\ L_u(x, a)$

$Subject\ to$: $L_i \le x_i \le U_i, \qquad \forall v_i \in V$

$Min\_TT_i \le D_i \le Max\_TT_i, \forall v_i \in V$

$OC_i \le Unit\_cap \cdot x_i, \forall v_i \in V$

$\forall v_i \in V, i = 1,2, \dots, n$

$where\ u = (u_0, \dots, u_{n+s}), u_i = \sum_{j \in input(i)} \lambda_{ji}$ *for* $0 \le i \le n+s$, *and* $L_u(x) = \sum_{i=1}^{n+s} u_i D_i$

C. Solving LRS/u

With the modified Elmore delay model discussed in Section 5.3, we will see how to solve LRS/u with any fixed $u \ge 0$.

$$L_u(x) = \sum_{i=1}^{n+s} u_i D_i$$

$$= \sum_{i=n+1}^{n+s} u_i D_i + \sum_{i=1}^{n} u_i D_i$$

$$= \sum_{i=n+1}^{n+s} u_i r_i C_i^{'} + \sum_{i=1}^{n} u_i \alpha_i \beta_i \cdot (r_i \cdot C_i^{'} + D_{i0})$$

For $1 \le i \le n$, with Eq (5. 4): $r_i = \hat{r}_i / x_i + r_{i0}$, we have:

$$L_u(x) = \sum_{i=n+1}^{n+s} u_i r_i C_i^{'} + \sum_{i=1}^{n} u_i \alpha_i \beta_i \cdot (\hat{r}_i / x_i + r_{i0}) \cdot C_i^{'} + \sum_{i=1}^{n} u_i \alpha_i \beta_i \cdot D_{i0}$$

$$= \sum_{i=n+1}^{n+s} u_i r_i C_i^{'} + \sum_{i=1}^{n} u_i \alpha_i \beta_i \cdot (\hat{r}_i / x_i + r_{i0}) C_i^{'} + \sum_{i=1}^{n} u_i \alpha_i \beta_i \cdot D_{i0} \qquad (5.\ 12)$$

For any $1 \le i \le n$, if $j \notin input(i)$, $u_j r_j C_j'$ is independent of x_i. If $j \in input(i)$, $u_j r_j C_j' = u_j \alpha_j \beta_j \cdot r_j \cdot \hat{c}_i x_i + terms\ independent\ of\ x_i$. So we have:

$$L_u(x) = \left(\sum_{j \in input(i)} u_j \alpha_j \beta_j r_j \hat{c}_i \right) \cdot x_i + \frac{u_i \alpha_i \beta_i \hat{r}_i C_i'}{x_i} + terms\ independent\ of\ x_i \quad (5.\ 13)$$

Then the optimal value of x_i is:

$$x_i^* = \sqrt{\frac{u_i \alpha_i \beta_i \hat{r}_i C_i'}{\left(\sum_{j \in input(i)} u_j \alpha_j \beta_j r_j \hat{c}_i\right)}} \quad (5.\ 14)$$

Then concerning the transition time constraints and output capacitance constraints, we have the following size constraints:

$$L_i \le x_i \le U_i$$

$$Min\_TT_i \le D_i \le Max\_TT_i$$

$$OC_i \le Unit\_cap \cdot x_i$$

Considering the modified Elmore delay model with $D_i = \alpha_i \beta_i \{(\hat{r}_i / x_i + r_{io}) \cdot C_i' + D_{i0}\}$, we have

$$\frac{\hat{r}_i}{(Max\_TT_i/\alpha_i\beta_i - D_{i0})/C_i' - r_{io}} \le x_i \le \frac{\hat{r}_i}{(Min\_TT_i/\alpha_i\beta_i - D_{i0})/C_i' - r_{io}}$$

and

$$x_i \ge OC_i / Unit\_cap_i$$

So combing all the constraints, we set new lower bound L_i^* and upper bound U_i^* as:

$$L_i^* = max\left\{L_i,\ OC_i/Unit\_cap_i, \frac{\widehat{r_i}}{(Max\_TT_i/\alpha_i\beta_i - D_{i0})/C_i' - r_{io}}\right\} \quad (5.\ 15)$$

and

$$U_i^* = min\left\{U_i, \frac{\widehat{r_i}}{(Min\_TT_i/\alpha_i\beta_i - D_{i0})/C_i' - r_{io}}\right\} \quad (5.\ 16)$$

And combing the Eq. (5. 14), (5. 15) and (5. 16), then the optimal local resizing of component i is given by the following equation:

$$x_i^* = min\left\{U_i^*, max\left(L_i^*, \sqrt{\frac{u_i\alpha_i\beta_i\widehat{r_i}C_i'}{(\sum_{j\in input(i)} u_j\alpha_j\beta_j r_j\widehat{c_i})}}\right)\right\} \quad (5.\ 17)$$

As proved by Chen et al. in [10], the algorithm to solve LRS/u always converges and the solution is optimal to LRS/u.

Hence, with a fixed $u \geq 0$, we can solve LRS/u by an iterative and greedy algorithm which resizes the gate components. Figure 5. 3 outlines the algorithm, denoted as $SOLVE\_LRS/u$. Initially, we set all the gates with minimum sizes. For each iteration, we adjust the gates one at a time. Then each time, we only adjust one gate based on Eq.(5. 17), and keep all the other gates fixed. Thus, we exam all the gates until they converge to the optimal sizes. Obviously, each iteration take $O(n)$ time. And Chen et al. [10] proved that the complexity of $SOLVE\_LRS/u$ is $O(n)$.

Algorithm $SOLVE\_LRS/u$

Input: $u = (u_0, \dots, u_{n+s})$, and all gate information

Output: $X = (x_1, \dots, x_n)$, which minimizes $L_u(X)$

1. **Repeat**
2. Update timing information
3. For each gate $i, where\ 1 \le i \le n$
4. $$L_i^* = max\left\{L_i,\ \frac{OC_i}{Unit_{cap_i}},\ \frac{\widehat{r_i}}{(Max\_TT_i/\alpha_i\beta_i - D_{i0})/C_i' - r_{io}}\right\}$$
5. $$U_i^* = min\left\{U_i,\ \frac{\widehat{r_i}}{(Min\_TT_i/\alpha_i\beta_i - D_{i0})/C_i' - r_{io}}\right\}$$
6. $$x_i^* = min\left\{U_i^*, max\left(L_i^*, \sqrt{\frac{u_i\alpha_i\beta_i\widehat{r_i}C_i'}{(\sum_{j\in input(i)} u_j\alpha_j\beta_j r_j\widehat{c_i})}}\right)\right\}$$
7. **Until** no further improvement

Figure 5. 3. Pseudo code for $SOLVE\_LRS/u$ algorithm

D. Adjust λ

In this part, we represent how to adjust the Lagrangian multipliers for maximizing LRS/λ.

Chen et al. [10] introduced the method called subgradient-based direction, which calculates the subgradient direction by adding the product of step size ρ_k and the difference between the current arrival time and the expected arrival time. However, this method is not well effective and may easily lead to negative λ in some cases. Zhou et al. [40] introduced

a subgradient method to update the multipliers according to the percent of the current arrival time to the expected arrival time. And we further refined it so that it has different updates depending on its locations in the circuit. We define the following equations to update multipliers λ_{ji}, where $j \in input(i)$:

$$\lambda_{ji} = \begin{cases} \lambda_{ji} \times (\frac{a_j}{a_0})^{\rho_k} & if\ i = 0 \\ \lambda_{ji} \times (\frac{a_j + D_i}{a_i})^{\rho_k} & if\ 1 \le i \le n \\ \lambda_{ji} \times (\frac{D_i}{a_i})^{\rho_k} & if\ n+1 \le i \le n+s \end{cases} \quad (5.\ 18)$$

We have outlined the pseudo code for delay minimization by Lagrangian relaxation (DMLR) in Figure 5. 4.

Algorithm DMLR

Input: Circuit and library information

Output: Solution X with minimum circuit delay

1. Set initial size as minimum cell size
2. **Repeat**
3. Update timing information
4. Adjust λ with Eq. (5. 18)
5. Project λ to satisfy Kuhn-Tucker conditions:

$$1 = \sum_{j \epsilon input(0)} \lambda_{j0}$$

$$\sum_{i \epsilon output(k)} \lambda_{ki} = \sum_{j \epsilon input(i)} \lambda_{ij}$$

6. Call SOLVE_LRS/u to find the X
7. **Until** no further improvement

Figure 5. 4. Pseudo code for DMLR algorithm

5.5 Modified Nearest Rounding and Post Power Optimization

In this section, we present our algorithm to project continuous solution to discrete solution and keep reducing extra delay cost during nearest rounding. And we apply the post power optimization afterwards to further reduce the power consumption.

Our algorithm presented in Section 5.4 gives us an optimal continuous solution. After getting the continuous solution, we usually apply nearest rounding to project to discrete solution. However, during the nearest rounding, it is a hard to tell whether we should round up or round down the continuous size, since it may introduce extra cost when the continuous solution is either rounded up or rounded down. Then we propose a quick procedure to reduce the extra delay cost by considering resizing the gates with either one size up or one size down. And we use the local discrete delay difference introduced in Chapter 4 to calculate delay cost, which is much more time efficient.

The pseudo code of Modified Nearest Rounding (MNR) is outlined in Figure 5. 5.

Algorithm MNR

Input: Continuous solution X with minimum circuit delay

Output: Discrete solution X'

1. Project continuous solution X to discrete solution X' using classic nearest rounding
2. Foreach gate i
3. $v_i.resizable = true$
4. **Repeat**
5. Update timing information
6. Foreach gate i
7. If $v_i.resizable == true$
8. Find $\Delta D_{max}(i)$
9. $x_i = x_i^{'}$, and $v_i.resizable = false$
10. **Until** no further improvement

Figure 5. 5. Pseudo code for Modified Nearest Rounding algorithm

Our entire algorithm of discrete gate sizing with Lagrangian relaxation (DSLR) is outlined in Figure 5. 6.

Algorithm DSLR

Input: Circuit and library information

Output: Solution $X^{*\prime}$ with minimum circuit delay and optimized leakage power consumption

1. Call DMLR to find the continuous solution X with minimum circuit delay
2. Call MNR to project continuous solution X to discrete solution X'
3. Apply post power optimization to further reduce power consumption

Figure 5. 6. Pseudo code for discrete gate sizing with Lagrangian relaxation

5.6 Experimental Results

We implemented the algorithm in C++ and the experiments are performed on a 3.4 GHz Intel Core i-7 computer. We use the test circuits from ISCAS 87' benchmark, and the standard cell library provided by the ISPD 2012 Discrete Gate Sizing Contest [6].

Table 5. 5. Circuit Delay and Power Comparison between Our Algorithm and Discrete Optimal Solution. *Delays* are in *ps. Powers* are in *uW.*

Circuit	# Of Gate	Continuous Solution		Nearest Rounding with Adjustment		Post Power Optimization		Discrete Optimal Solution from Algorithm in Chapter 4		Comparison	
										Our Algorithm vs.Heuristics in Chapter 4	
		Delay	Power	Delay	Power	Delay	Power	Delay	Power	Delay	Power
C432	160	799	7577	759	6790	753	2152	694	2280	+8.4%	-5.6%
C499	202	473	9979	565	10432	565	4864	531	5100	+6.4%	-4.6%
C880	383	818	8863	885	6094	884	3340	840	3364	+5.2%	-1%
C1355	546	791	16422	878	12064	868	5072	936	5312	-7.2%	-4.5%
C1908	880	816	20092	1055	16846	1054	7104	1018	6824	+3.6%	+4.1%
C2670	1193	786	13213	1020	15408	1020	9154	994	9352	+2.6%	-2.1%
C3540	1669	1090	22344	1518	24446	1518	13230	1451	13056	+4.6%	+1.3%
C5315	2307	1031	28193	1447	32838	1446	20474	1316	20572	+9.9%	-0.05%
C6288	2416	2759	25688	4151	64552	4151	26216	4086	23520	+1.6%	+11.5%
C7552	3512	950	31584	1227	35452	1217	26608	1182	26600	+2.9%	0.0%
Avg.	1327	1031	18395	1350	22492	1348	11821	1305	11598	+3.8%	-0.1%

Table 5. 5 shows the delay and power in each stage of our algorithm. We listed the delay and power after continuous solution, nearest rounding with adjustment, and after post power optimization. And we compare those with the discrete optimial solution from the heuristic algorithm in Chapter 4. As we can see from the table that on average, the delay of our new algorithm is within 3.8% more than the minimum solution of the algorithm in Chapter 4. And the power is averagely 0.1% less than the optimal solution of the algorithm in Chapter 4, which proves that our algorithm is effective in calculating the minimum delay with optimized power.

Table 5. 6. Delay and Power Comparison between Our Algorithm (NEW) and Liu's Algorithm [36]. *Delays* are in *ps. Powers* are in *uW.*

Circuit	New		Liu [36]		New vs Liu [36]	
	Delay	Power	Delay	Power	Delay	Power
c432	752.6	2152	870.4	2684	-13.54%	-19.82%
c499	564.8	4864	627.2	3664	-9.95%	32.75%
c880	884.0	3340	1052.7	4574	-16.02%	-26.98%
c1355	868.4	5072	960.5	5176	-9.59%	-2.01%
c1908	1054.5	7104	1074.4	6246	-1.86%	13.74%
c2670	1020.0	9154	1068.3	9388	-4.52%	-2.49%
c3540	1517.5	13230	1687.9	17618	-10.09%	-24.91%
c5315	1446.4	20474	1338.9	23904	8.03%	-14.35%
c6288	4150.6	26216	3837.4	30484	8.16%	-14.00%
c7552	1216.7	26608	1227.9	27450	-0.91%	-3.07%
Average	1347.5	11821.4	1374.6	13118.8	-5.03%	-6.11%

And to validate the effectiveness of our algorithm, we also compared our algorithm with the previous work from Liu and Hu [36], who proposed a method that employs bi-directional solution search. However, the method cannot predict the effect of cell change on its input side, which causes imprecision in searching and pruning solution. We implemented the JRR from [36] to obtain the minimum delay solution first and then minimize power with the minimum delay constraint. As the data Table 5. 6, we can see the delay and power of our algorithm is 5.03% and 6.11% less than that of [36].

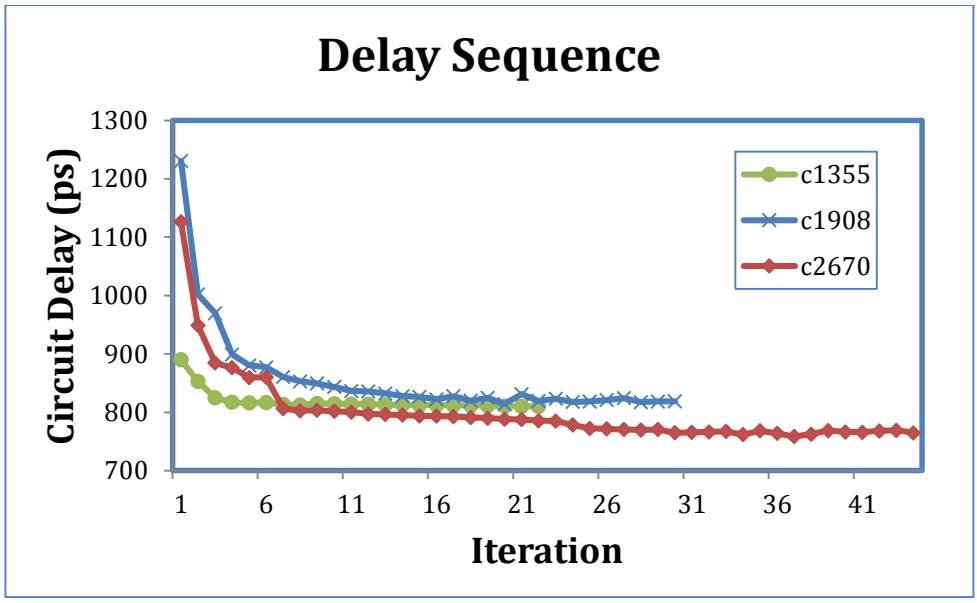

Figure 5. 7. Delay sequence of circuit c1355, c1908 and c2670.

Delays are in *ps*.

Figure 5. 7 shows the delay sequence of circuit c1355, c1908 and c2670. As we can see, the delay decreases as the iterations go on, and finally converges to the optimal solution. And it usually takes more iterations to converge for circuits with more gates.

To compare the runtime performances, Figure 5. 8 shows the runtime versus the number of gates. And polynomial curve is plotted for comparison.

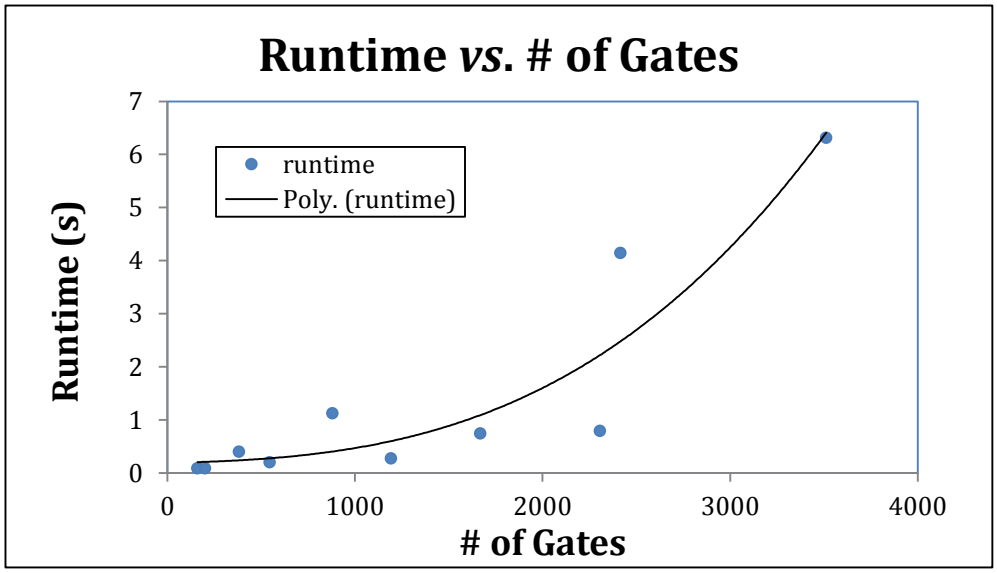

Figure 5. 8. Runtime comparison vs. # of gates.

Runtimes are in *seconds(s).*

5.7 Summary

In this work, we have modified the classic Elmore delay model so it fits the features of the lookup tables from ISPD 2012 standard cell library. And we have presented a new algorithm to solve the discrete gate sizing problems by reusing the Lagrangian relaxation method, which was previously used to solve the continuous gate sizing problems only. We have presented how to use the modified Elmore delay model to minimize circuit delay for combinational circuit using the revised Lagrangian relaxation approach. And we also take consideration of the input transition time constraints and output load capacitance constraints. The Lagrangian relaxation approach is composed of two subproblems, denoted as the Lagrangian relaxations subproblem and the Lagrangian dual problem. With the Kuhn-Tucker conditions, we can solve the Lagrangian subproblem with an iterative algorithm, which is approved to converge to the global optimal. The classical subgradient optimization method is used to update the Lagrangian multipliers.

The experimental results show that the revised Lagrangian relaxation method with the modified Elmore delay model is able to achieve the minimum delay within 4% more of the optimal values.

Chapter 6

Conclusion

6.1 Summary

In this work, we have a thorough review of the gate sizing problem, which evolved from continuous solutions to discrete solutions, and it has been studied with different methods and techniques, which are applied to solve the delay, area, power consumption etc. However, the existing works all have their limitation; some are dedicated to solving unrealistic continuous problems, and others can only handle outdated simple delay model. Hence, all the limitations challenge us to explore better ways to solve the problem.

In this work, we have proposed a series of discrete sizing methodologies which can be applied to improve circuit performance with different optimization concerns and deal with different modern gate models.

In Chapter 2, we have proposed an algorithm which directly solves the discrete gate sizing problem with delay minimization. We use local delay difference to evaluate the delay cost when selecting cell replacement, which is much faster than global search. And another great thing about the local search is that it is able to avoid trapped at the local minima and reach the highly satisfactory global solution. And our algorithm is only within 4.9% more than the optimal continuous solution.

In Chapter 3, we have proposed a method to optimize the circuit area while maintaining the minimum circuit delay. We also apply the local delay difference in

searching for appropriate smaller cells locally, which takes much less time compared with that uses global search.

In Chapter 4, we have proposed a two-phase optimization method, which is able to minimize the circuit delay in phase 1 and reduce the power consumption in phase 2. This method works with the lookup tables from standard cell library. A unique part of this method is that it is able to handle the constraint of input slews and output load capacitance, which no existing techniques can do now.

In addition to algorithms, we also deal with the problem from the modeling aspect in Chapter 5. We modified the classic Elmore delay model so that it not only captures the features of the standard cell library model, but also keeps its original simplicity at the same time. With our modified new gate model, we are able to solve the discrete problem using continuous Lagrangian relaxation, which was thought to solve continuous sizing problem only. But now, with our new model, it is easier to solve and implement other discrete sizing problems.

As a conclusion of this work, we have proposed a series of methods which can be applied to solve the discrete gate sizing problems with different optimization aims. They are able to deal with modern accurate gate models, like lookup tables from standard cell library. And we used the latest standard cell library information for experiments and comparisons. Compared with other existing works, our methods are more flexible and robust as they can deal with more complicated models and it takes much less time to run. And we contribute to the discrete gate sizing field by providing our experiments, and we welcome any advice, judgment and critics.

6.2 Suggestions for future work

We may suggest continuing the research of discrete gate sizing from the following main aspects:

1. Circuit optimization with double-sided delay constraint

   In our current work, the circuit delay only has one maximum delay constraint, as an upper bound of delay constraint, and we allow the circuit delay to be as minimum as it could be. However, sometimes the circuit delay is required to be more than a certain amount of time. For example, the signal should hold up more than the hold time. So we may set a minimum delay constraint for the circuit, as the lower bound of delay constraint. Thus, we may improve our work so that it can deal with both maximum (upper bound) and minimum delay (lower bound) constraint, or we call it as *double-sided delay constraint*.

   To solve this problem, our initial idea is to define the gates on the shortest path, as the delay of the shortest path determines the shortest delay. And we may applied the algorithm proposed in Chapter 2, Chapter 3 and Chapter 4 to search for the best cell replacement on the shortest path and optimize the circuit delay, area or power consumption.

   However, some gates may be on both the critical path and shortest path, and the change of their sizes may violate either of the constraint, which makes it more complicated than the one-sided constraint. Thus, we need to see how to solve this problem and make it work for the double-sided constraints.

2. Separate delay constraint at output nodes

In this dissertation research work, we assume all the output nodes in the circuit has the same delay constraints, and we use a pseudo output node connecting all the output nodes, thus we can easily define the critical path by starting from the pseudo output node and trace back to the input nodes.

In the future, we may want to separate the delay constraint of each output node, which is apparently more versatile for circuit design. But the challenge is that it may be hard to define the critical path if each output node has different delay requirement. And since our proposed methods are critical-path based sizing algorithms, we may need to do further work on that part.

3. Combine threshold voltage assignment in gate selection

   There are many existing works optimizing leakage power consumption using threshold voltage assignment. And surely we can also combine the threshold voltage assignment into our methods.

4. Keep refining the gate model with the modern cell library

   We will keep refining our proposed gate model in Chapter 5 so that it will always stay updated with the features of new modern cell library in the future. And we need to keep refining our method with the development of the VLSI industry.

In addition to the aspects above, since the VLSI industry will keep improving, there always will be new issues and concerns coming out for gate sizing problems in circuit design, such as more accurate gate models, or other versatile timing constraints, etc. We always need to keep refining our methods so that it will keep updated with the development of the industry.

Bibliography

[1] J. H. Lee., “Implications of Modern Semiconductor Technologies”. Ph.D. dissertation, University of California Los Angeles, 2012.

[2] W. C. Elmore, “The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers,” *Journal of Applied Physics*, January 1948, Volume 19, Issue 1, pp. 55-63.

[3] P. Penfield and J. Rubenstein, “Signal delay in RC tree networks,” *Proceedings of the 19th Design Automdion Conference,* 1981.

[4] R. Gupta, B. Tutuianu, and L.T. Pileggi. “The Elmore delay as a bound for RC trees with generalized input signals,” *IEEE Trans. on Computer-Aided Design*, 16(1):95–104, 1997.

[5] L. T. Pillage and R. A. Rohrer, “Asymptotic Waveform Evaluation for Timing Analysis,” *IEEE transactions on Computer-Aided Design*. vol. 9, No. 4, April 1990.

[6] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, C. Zhuo, "The ISPD -2012 Discrete Cell Sizing Contest and Benchmark Suite," *Proc. ACM International Symposium on Physical Design*, pp. 161-164, 2012.

[7] A.E. Ruehli, P.K. Wolff, and G. Goertzel. “Analytical power/timing optimization technique for digital system,” In *Proc. Design Automation Conference*, pp. 142–146, 1977.

[8] J. Fishburn and A. Dunlop, "Tilos: A polynomial programming approach to transistor sizing," in *Proc. Int. Conf. Comput – Aided Des.,* 1985, pp. 326-328.

[9] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang. "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Trans. on Computer-Aided Design*, vol. 12, pp. 1621–1634, Nov. 1993.

[10] C.-P Chen, C. Chu, and D. Wong, "Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation," *IEEE Trans. Comput.-Aided Design Intergr. Circuits Syst.,* vol. 18, no. 7, pp. 1014-1025, Jul. 1999.

[11] J. Wang, D. Das and H. Zhou, "Gate sizing by Lagrangian relaxation revisited," in *2007 IEEE/ACM International Conference on Computer-Aided Design*, pp. 111-118.

[12] T. Karnik, Yibin Ye, J. Tschanz, Liqiong Wei, S. Burns, V. Govindarajulu, V. De, and S. Borkar. "Total power optimization by simultaneous dual-Vt allocation and device sizing in high performance microprocessors," In *Proc. Design Automation Conference*, pp. 486–491, 2002.

[13] H. Tennakoon and C. Sechen, "Gate sizing using lagrangian relaxation combined with a fast gradient-based pre-prossing step," in *Proc. ICCAD*, 2002, pp. 395-402.

[14] K. Kasamsetty, M. Ketkar, and S.S. Sapatnekar. "A new class of convex functions for delay modeling and its application to the transistor sizing problem [CMOS gates]," *IEEE Trans. Comput.-Aided Design Integ. Circuits Syst.*, vol. 19, no. 7, pp. 779–788, Jul. 2000.

[15] S. P. Boyd, S.J. Kim, D. D. Patil, and M. A. Horowitz. "Digital dircuit optimization via geometric programming," *Operations Research,* 53(6):899, 2005.

[16] M. R. C. M. Berkelaar and J. A. G Jess. “Gate sizing in MOS digital circuits with linear programming,” in *Proc. DATE*, 1990, pp. 217-221.

[17] D. Chinnery and K. Keutzer, “Linear programming for sizing, vth and vdd assignment,” in *Proc. ISLPED*, 2005, pp. 149-154.

[18] J. Lee and P. Gupta, “Incremental gate sizing for late process changes,” in *Proc. ICCD*, 2010, pp. 215-221.

[19] H. Chou, Y.H. Wang, and C. C. P. Chen. “Fast and effective gate-sizing with multiple-Vt assignment using generalized Lagrangian Relaxation,” In *Proc. ASPDA*, pp. 381–386, 2005.

[20] S. Roy, W. Chen, C.P. Chen, and Y.H. Hu. “Numerically convex forms and their application in gate sizing,” *IEEE Trans. Comput.-Aided Des.*, vol. 26, no. 9, pp. 1637–1647, Sep. 2007.

[21] N. Menezes, R. Baldick, and L. T. Pileggi, “A sequential quadraticc programming approach to concurrent gate and wire sizing,” *IEEE Trans. Compu.-Aided Des.*, vol. 16, no. 8, pp. 867-881, Aug. 1997.

[22] H. Ren and S. Dutt, “A network-flow based cell sizing algorithm,” in *Proc. Int. Workshop Logic Syn*., 2008, pp. 7-14.

[23] S. Shah, A. Srivastava, D. Sharma, D. Sylvester, D. Blaauw, and V. Zolotov, “Discrete Vt assignment and gate sizing using a self-snapping continuous formulation,” In *Proc. Int. Conf. Computer-Aided Design*, pp. 705–712, 2005.

[24] S. Hu, M. Ketkar, and J. Hu, “Gate sizing for cell-library-based designs,” *IEEE Trans. Comput.-Aided Des.*, vol. 28, no. 6, pp. 818-825, Jun. 2009.

[25] M. Rahman, H. Tennakoon, and C. Sechen, “Power reduction via near optimal library-based cell-size selection,” in *Proc. DATE*, 2011, pp. 1-4.

[26] W. -N. Li, “Strongly NP-hard discrete gate sizing problems,” in *Proc. ICCD*, 1993, pp. 468-471.

[27] P. K. Chan, “Algorithms for library-specific sizing of combinational logic,” in *Proc. DAC*, 1990, pp. 353-356.

[28] Stephan Held. “Gate sizing for large cell-based designs,” In *Proc. DATE*, pp. 827–832, 2009.

[29] S. Sirichotiyakul, T. Edwards, C. Oh, J. Zuo, A. Dharchoudhury, R. Panda, and D. Blaauw. “Stand-by power minimization through simultaneous threshold voltage selection and circuit sizing,” In *Proc. Design Automation Conference*, pp. 436–441, 1999.

[30] L. Wei, K. Roy, and C. K. Koh. “Power minimization by simultaneous dual-Vth assignment and gate-sizing,” In *Custom Integrated Circuits Conference, 2000. CICC. Proceedings of the IEEE 2000*, pp. 413–416, 2000.

[31] S. Sirichotiyakul, T. Edwards, C. Oh, R. Panda, and D. Blaauw. “Duet: An accurate leakage estimation and optimization tool for dual-Vt circuits,” *IEEE Trans. on Very Large Scale Integration (VLSI)*, 10(2):79–90, 2002.

[32] A. Srivastava, D. Sylvester, and D. Blaauw, "Power minimization using simultaneous gate sizing, dual-Vdd and dual-Vth assignment," In *Proc. ACM Design Automation Conf.*, pp.783-787, 2004.

[33] Puneet Gupta, Andrew B. Kahng, Puneet Sharma, and Dennis Sylvester. "Selective gate-length biasing for cost-effective runtime leakage control," In *Proc. Design Automation Conference*, pp. 327–330, 2004.

[34] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *IEEE Trans. VLSI Syst.*, vol. 5, no. 4, pp. 465-472, Dec. 1997.

[35] O. Coudert, R. Haddad, and S. Manne. "New algorithms for gate sizing: a comparative study," In *Proc. Design Automation Conference*, pp. 734–739, 1996.

[36] Y. Liu and J. Hu. "A new algorithm for simultaneous gate sizing and threshold voltage assignment," In *Proc. Int. Conf. Physical Design*, pp. 27–34, 2009.

[37] Y.L. Huang, J. Hu, andW. Shi. "Lagrangian relaxation for gate implementation selection," In *Proc. Int. Conf. Physical Design*, pp. 167–174, 2011.

[38] M. M. Ozdal, S. Burns, and Jiang Hu. "Gate sizing and device technology selection algorithms for high-performance industrial designs," In *Proc. Int. Conf. Computer-Aided Design*, pp. 724–731, nov. 2011.

[39] V. S. Livramento, C. Guth, J. L. Guntzel and M. O. Johann, "Fast and efficient Lagrangianrelaxation-based discrete gate sizing," *Proceedings of the conference on Design, Automation and Test in Europe. EDA Consortium*, 2013.

[40] S. Zhou, H. Yao, Q. Zhou, and Y. Cai, "Minimization of circuit delay and power through gate sizing and threshold voltage assignment," in *Proc. ISVLSI*, 2011, pp. 212-217.

[41] G. Flach, T. Reimann, G. Posser, M. Johann, R. Reis, "Effective method for simultaneous gate sizing and Vth assignment using Lagrangian relaxation," *IEEE*

Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp 546-557, April 2014.

[42] Hamed Abrishami, Jinan Lou, Jeff Qin, Juergen Froessl, and Massoud Pedram, "Post Sign-off Leakage Power Optimization," In *Proc. Design Automation Conference*, 2011.

[43] W. Chuang, S.S. Sapatnekar, and I.N. Hajj. "Timing and Area Optimization for Standard-Cell VLSI Circuit Design," *IEEE Trans. on Computer-Aided Design*, vol. 14, no. 3, pp.308-320, 1995.

[44] S. Boyd, S. Kim, D. Patil, and M. Horowitz, "Digital circuit optimization via geometric programming," *Oper: Res.*, vol. 53, no. 6, pp. 899-932, Nov./Dec. 2005.

[45] N. Hanchate and N. Ranganathan, "Simultaneous interconnect delay and cross talk noise optimization through gate sizing using game theory," *IEEE Trans. Comput*., vol. 55, no. 8, pp. 1011-1023, Aug. 2006.

[46] K.Jeong, A. B. Kahng and H. Yao, "Revisiting the Linear Programming Framework for Leakage Power vs. Performance Optimization," *Proc. ISQED*, pp. 127-134, 2009.

[47] I. Lin and Y. Chang, "An efficient algorithm for statistical circuit optimization using Lagrangian relaxation," in *2007 IEEE/ACM International Conference on Computer-Aided Design*, pp. 119-124.

[48] S. Roy, Y. H. Hu, C. C. Chen, S. Hung, T. Chiang and J. Tseng, "An optimal algorithm for sizing sequential circuits for industrial library based designs," in 2008 *Asia and South Pacific Design Automation Conference*, pp. 148-151.

[49] W. Chuang, S. S. Sapatnekar and I. N. Hajj, "Delay and area optimization for discrete gate sizes under double-sided timing constraints," in *Proceedings of the IEEE 1993 Custom Integrated Circuits Conference*, pp. 9-4.

[50] dos S Livramento, V., et al. "Lagrangian relaxation-based Discrete Gate Sizing for leakage power minimization," *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on. IEEE,* 2012.

[51] I-J. Lin, T.-Y. Ling, and Y.-W. Chang, "Statistical Circuit Optimization Considering Device and Interconnect Process Variations," *Proc. SLIP*, pp. 47-54, Mar. 2007.

[52] M. Mani, A. Devgan, and M. Orshansky, "An Efficient Algorithm for Statistical Minimization of Total Power under Timing Yield Constraints," *Proc. DAC*, pp. 307-314, Jun. 2005.

[53] M. M. Ozdal, S. Burns, and J. Hu, "Algorithms for gate sizing and device parameter selection for high-performance designs," *IEEE Trans. On CAD,* vol. 31, pp. 1558-1571, Oct 2012.

[54] D. Nguyen, A. Davare, M. Orshansky, D. Chinnery, B. Thompson, and K. Keutzer, "Minimization of dynamic and static power throught joint assignment of threshold voltages and sizing optimization," in *Proc, ISLPED, 2003*, pp. 158-163.

[55] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, Inc., second edition, 1993.

VITA

NAME OF AUTHOR: Jiani Xie

PLACE OF BIRTH: Shanghai, People's Republic of China

DATE OF BIRTH: November 12, 1986

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

Syracuse University, Syracuse, New York, the United States

Fudan University, Shangahi, People's Republic of China

DEGREES AWARDED:

Master of Science in Computer Engineering, 2011, Syracuse University

Bachelor of Science in Communications Science and Engineering, 2008, Fudan University

PROFESSIONAL EXPERIENCE:

Teaching Assistant, Department of Electrical and Computer Engineering, Syracuse University, 2008 - 2013