國立交通大學

電子工程學系 電子研究所碩士班

碩 士 論 文

考慮緩衝器的躍遷時間之低功率時鐘樹生成

Low Power Buffered Clock Tree Synthesis Considering Buffer Transition Time

研 究 生：陳皇良

指導教授：陳宏明 博士

中 華 民 國 九十四 年 六 月

考慮緩衝器的躍遷時間之低功率時鐘樹生成

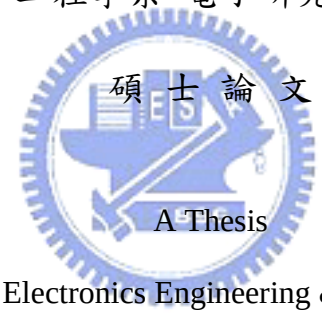
Low Power Buffered Clock Tree Synthesis Considering Buffer Transition Time

研 究 生：陳皇良　　Student：Huang-Liang Chen

指導教授：陳宏明　博士　　Advisor：Hung-Ming Chen

國 立 交 通 大 學

電子工程學系 電子研究所碩士班



碩 士 論 文

A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIEMCE

in

Electronics Engineering

June 2005

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 四 年 六 月

考慮緩衝器的躍遷時間之低功率時鐘樹生成

學生: 陳皇良　　指導教授: 陳宏明 博士

國 立 交 通 大 學

電 子 工 程 學 系　電 子 研 究 所 碩 士 班

摘要

隨著製造技術的進步，有百萬個或千萬個閘在電路中，這使得時鐘網路設計上會有很大的挑戰，由於電路頻率愈來愈高，所花費的功率也越來越多，且大部分都在時鐘樹本身，如沒有小心的規劃時鐘網路，電路將承受高功率的損耗。在本篇論文中，我們發展出一個方法能夠在時鐘樹合成時，達到低功率的效果。此方法基本上根據分析任意提供的緩衝器資料庫，進一步發現最佳的躍遷時間來有效的插入緩衝器，達到低功率的成果。由實驗結果可以得知，我們的方法能有效的降低功率。跟之前的利用調整閘大小的方法，我們能夠減少平均百分之十以上的功率損失而且較小的非對稱時鐘差。

Low Power Buffered Clock Tree Synthesis Considering Buffer Transition Time

Student: Huang-Liang Chen Advisor: Dr. Hung-Ming Chen

Department of Electronic Engineering
Institute of Electronics
National Chiao Tung University

Abstract

Clocking has been playing a very important role in current VLSI designs. As technology advances, there exist millions or billions of gates on a chip, which makes the clock network design much more challenging since synchronization in chip is one of the primary concerns. However, chips running at higher frequency consume much more power, mostly on clock distribution. Without carefully planning clock network, the chips will suffer from high power dissipation. In this thesis, we develop a methodology which can be applied in buffered clock tree synthesis to achieve low power demands. It is based on the analysis of any given buffer library in finding best transition time for low power and customized buffer insertion. The experimental results are encouraging. We have obtained average up to 10% power saving and smaller clock skew compared with a previous work based on gate sizing.

誌　　謝

這篇論文能順利完成，首先感謝指導老師陳宏明教授，在短暫的兩年中給予的啟蒙，讓我在 EDA 領域中有了初步的了解，也教導我正確的研究精神與態度，老師在研究討論中給予的精闢見解，對我有相當的影響，同時也感謝黃世旭教授，江蕙如教授，王俊堯教授，在口試期間中，細心的審閱論文並提供些寶貴意見，使論文能夠更加完整和嚴謹。

在研究工作上，必須要感謝實驗室的同學們，宗達、力中、雙議、冠中、敏菁、加易，在研究時的討論與學習，還有學弟們都協助，使在我研究所的過程中，留下了深刻的回憶。

更要感謝我的父母及家人，有你們的全力支持與關懷鼓勵讓我無後顧之憂，能夠全心全意專注在學業上，最後謹以此論文獻給所以幫助過我的朋友以及我所深愛的家人。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Clock designs play an important role in modern VLSI designs. As technology advances, a chip may have millions of gates with a very complex structure. The synchronization of clocks on a chip is critical to the performance and reliability of the chip. In a synchronous digital system, the clock signal defines the time reference for the movement of data within the system. Therefore the clock distribution need more careful design methodology in modern VLSI.

1.1 Motivation for Designing Low Power Clock Network

Due to a large amount of fan-outs that distribute over long routing distances in clock tree, among clock network designs, the buffered clock tree structure (as shown in Figure 1.1) is one of the most popular clock network designs adopted in modern VLSI designs. The buffered clock tree is a clock tree which has some buffers inserted between the clock source and all the registers driven by the clock signals. The major advantage of the buffered clock tree is that the buffered clock tree can maintain the

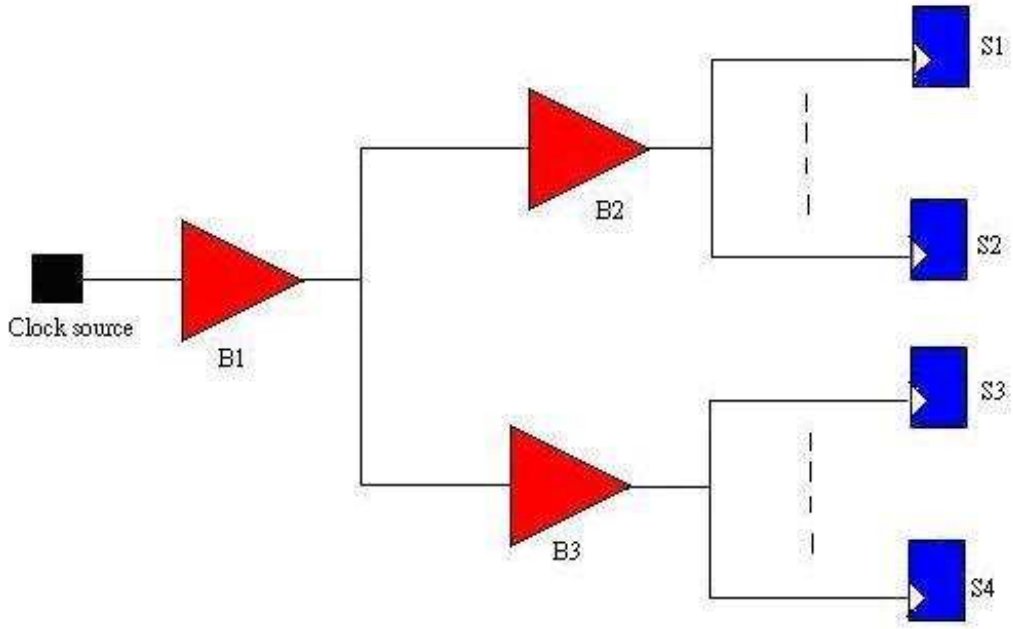


Figure 1.1: A buffered clock tree with synchronizing elements {S1,S2,S3,S4} and buffers{B1,B2,B3}.

quality of the clock waveform. Another benefit of buffered clock tree is that the inserted buffer in a clock path help reduce the interconnect resistance between clock source and driven register so that the wire RC delay is decreased. In order to meet timing constraint, we need to insert a lot of buffers along the clock paths. As a consequence, clock nets have a lot of buffers that may consume much more power, it is necessary to reduce the clocking power in high performance designs. In this thesis, we propose an effective algorithm to reduce the power consumption of buffered clock tree and we actually obtain up to 13% power saving compared with previous works based on gate sizing [7] [18], using the benchmarks from industry [1].

1.2 Organization of The Thesis

This thesis is organized as follows. In Chapter 2, we discuss some previous works in clock tree design. In particular, we introduce one greedy based algorithm to insert buffers for low power design from [18] and also introduce one proposed algorithm from [7] for get sizing. In Chapter 3, we formulate the problem of low power clock tree synthesis and introduce our estimation models. In Chapter 4, we will present our approaches on how to reduce power consumption by some observations, including buffer library analysis. Then we will show the experimental results in Chapter 5. Finally, we conclude the thesis with discussions for future work in Chapter 6.

Chapter 2

Historical Clock Tree Construction

In this chapter, we give a brief overview of clock tree design. In particular, we will describe the buffer sizing algorithm for low power clock network design from [7] and [18], and explain the implicit defects.

2.1 Previous Works

Previous works on clock tree construction focused on zero or near zero-skew routing. H-tree [2] is the earliest research about clock tree synthesis. Although H-tree structure can decrease clock skew, it has two constraints. One is that all clock sinks have the same loads. The other is that all sinks must be placed symmetrically. MMM(method of means and medians) algorithm [11] recursively partitions clock sinks and balances them with different length of interconnect to reduce clock skew. Tsay [16] is the first paper to probe into the exact zero skew clock tree construction. The exact zero skew algorithm recursively links the zero skew sub-tree. However, the algorithm will produce longer length of interconnect than other approaches do. Notably, the Deferred-Merge Embedding(DME) algorithm [5] [4] [3] achieve mini-

mum wire-length and tree radius for any given topology. It embeds internal nodes of a topology G via two phases. First, DME bottom-up constructs a tree of merging segments, or merging tree, and represents loci of possible placements of internal nodes in the zero-skew tree. Second, it top-down determines exact locations for the internal nodes of G. [12] proposes a buffered clock tree synthesis applying a clustering algorithm to obtain clusters of approximately equal capacitance loading. The skew due to load imbalance is minimized by wire sizing and wire lengthening. [9] investigates the problem of improving the performance of a synchronous digital system by adjusting the path delays of the clock signal from the central clock source to individual flip-flops. In addition, it proposes two linear programs. One minimizes the clock period while avoiding clock hazards; the other maximizes the minimum safety margin against clock hazard for a given period.

Until recently, it is getting more important to reduce the power consumed by the clock network due to higher frequency operation in modern digital systems. For the low power clock, some researches [15] [18] have tried to insert buffers to minimize the clock power. [7] proposes to use gate sizing in achieving power optimization. Authors of [6] and [8] exploit similarities in the switching activity of the clocked modules to reduce the number of the gates for clock gating. Tellez et al. in [17] investigate the problem of computing a lower bound on the number of buffers required in the clock tree given a maximum transition time constraint. Moreover, [13] [14] emphasizes that the transition time becomes the key factor of low power clock design and the tradeoff between the power and transition time when optimizing the clock tree is discussed. However, none of them can effectively use buffer library to help reduce power dissipation while inserting buffers in clock

network. Specifically, we introduce two works in gate sizing and low power clock network design in the following subsections.

2.2 Gate Sizing Algorithm in Reducing Power Consumption under Skew Constraint

Here we introduce buffer sizing algorithm (proposed in [7]) for power minimization and we modify it so that it can optimize the circuit under the skew constraint. Let a move represent a single gate resizing, which consists of replacing a nodes gate g_0 with an equivalent gate g_1. Let us denote $\Delta Cost$ (call "gradient") denote the variation $Cost_1 - Cost_0$ of some cost function $Cost$ produced by this move. In practice, although the change of buffer can affect the power of every buffer in the fanout and fanin cone, its effect on the power gradients decreases quickly. In other words, sizing a buffer has a large power effect in the first level of fanout but has a relatively smaller power effect on the second level of fanout and so on. Instead of evaluating the entire circuit, which is computationally expensive in an iterative algorithm, we only evaluate within a sub-clock tree composed of its parent and children of the change. We thus have a much cheaper gradient evaluation, and we avoid recomputing the gradients. This CPU time versus accuracy tradeoff has been experimentally validated in this algorithm.

Figure 2.1 shows the modified buffer sizing algorithm GS(Gate Sizing), which optimizes the clock tree power while conforming to the skew constraint. First, all buffers are put in an update list. All buffers in the list will be evaluated by the local cost function. The local cost function only calculates the immediate tree level of a buffer since evaluating the entire clock tree is too computationally expensive.

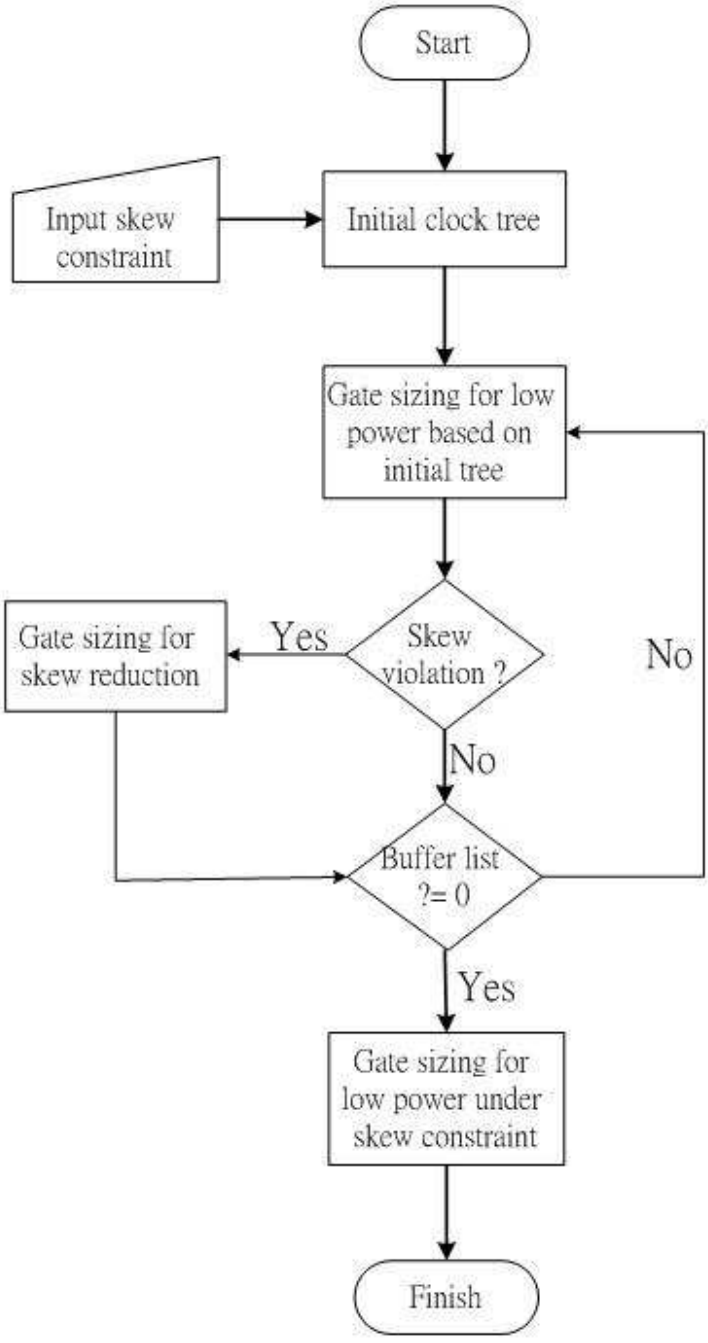


Figure 2.1: The design flow of gate sizing algorithm.

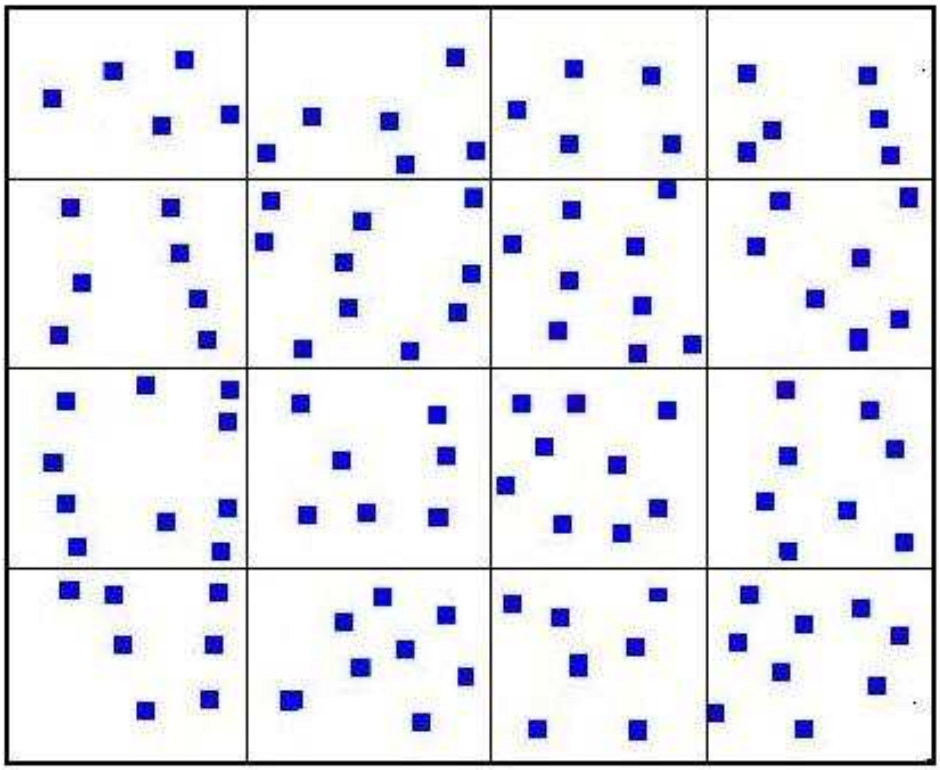


Figure 2.2: Sinks are partitioned into clusters with almost equivalent loading capacitances.

Therefore, we can evaluate the local cost function by trying all buffer sizes. After all buffers in the update list are evaluated by the local cost function, some buffers' size must be adjusted.

Since the buffer locations are limited gate sizing algorithm has the choices of the buffer types so we further change the buffer's locations for initial clock tree that the can cause different power consumption. Hence we briefly describe an intuitive clustering algorithm [10], which can change buffer's location for initial clock tree. We utilize this algorithm that creates N*N. It is shown in Figure 2.2 that each clus-

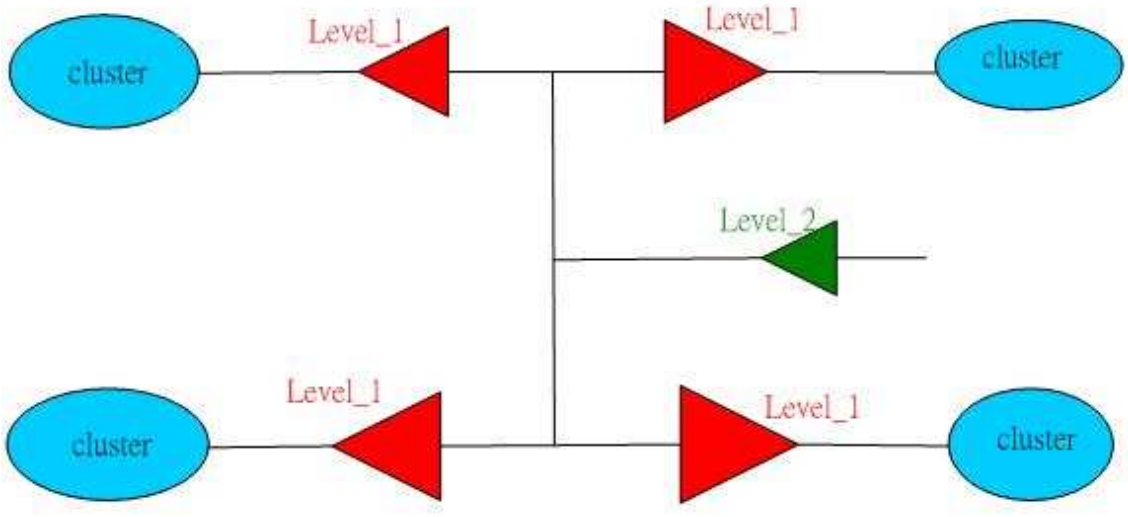


Figure 2.3: Recursive clock tree generation: buffer input capacitances and location used as input for next level of clustering.

ter has nearly equivalent loading capacitance. Each of these clusters is driven by identical buffers in the bottom level. For the next level of clustering, the coordinates of the buffers become the coordinates of the clock nodes, and the loading at these nodes is equal to the buffer input capacitance. Thus we general another level of the clock tree. The illustration is shown in Figure 2.3.

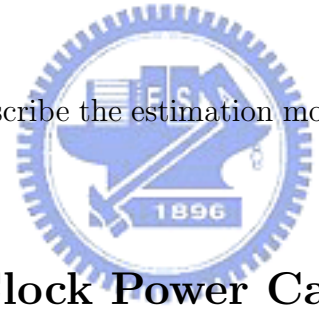
2.3 A Greedy Algorithm for Zero-Skew Low Power Buffered Clock Tree Construction

Here we introduce a greedy algorithm (proposed in [18]) on how bottom-up merging can handle buffer insertion and reliability issues while maintaining zero skew. In order to maintain the rise time constraint in each cell or to reduce interconnect

delay, we have to insert many buffers on the path from the root of clock tree to any sinks. The greedy algorithm chooses locally optimal solution by cost function, which is to decide where to insert buffer for low power designs. It may get good solution for low power design.

Chapter 3

Estimation Models for Low Power Buffered Clock Tree Synthesis and Problem Formulation



In this chapter, we briefly describe the estimation models and problem of low power buffered clock tree synthesis.

3.1 Delay and Clock Power Calculation

Clock tree synthesis is usually performed after cell placement to get more accurate physical information. The clock latency of a flip-flop's clock pin is the path delay starting from the root of clock tree, through the distribution cells, and ending at the clock pin (sink). In other words, the arrival time of a flip-flop's clock pin is its clock latency. Let all sinks of the circuit denote as $S=\{s_1,s_2,.....,s_n\}$. A clock tree T_S is an embedding of the connection topology in chip. The skew constraint of T_S is then defined as the maximum difference of clock latency, which is:

$$Skew(T_S) = max\{t(s_0, s_i) - t(s_0, s_j)\} \forall s_i, s_j \in S \quad (3.1)$$

where $t(s_i, s_j)$ is propagation delay from s_i to s_j and will be explained in the following subsection. We then present our power estimation model and formulate our problem.

3.1.1 Delay Estimation

Figure 3.1 illustrates a more abstract delay model. The time t needed for a signal to propagate from the input of a gate to the input of the next gate is t=t_d+t_o+t_c, where t_d is the propagation time inside the gate, t_o is the output transition time, and t_c is the connection time spent in the wire. The delay depends on the output load and on the input transition time.

The most accurate delay estimation is a simulation from differential equations, e.g., with SPICE, however it is too much CPU expensive. A table lookup based nonlinear interpolation delay model is within 3% of SPICE [7]. We therefore use such an accurate table lookup based nonlinear delay model.

For two-dimensional tables, the delay calculator uses a bilinear interpolation routine to determine these intermediate function values. For example, in Figure 3.2, we are to compute a cell's delay value D_{target} for a specific effective load C_{target} and input transition time T_r, delay calculator uses the delay values D_1,D_2,D_3 and D_4 of the four points in the table whose loading and input transition time come closest to the specified load and input slew values. To compute D_{target}, delay calculator determines $Temp_A$ and $Temp_B$ which are intermediate values computed by linear interpolation. The equations for delay calculator to calculate D_{target} are as follows.

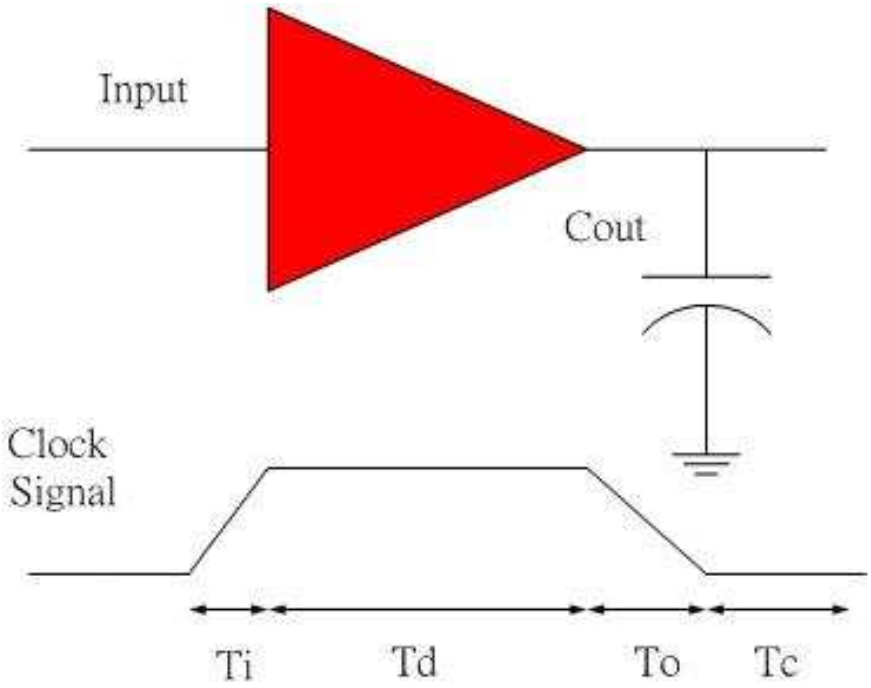


Figure 3.1: Input transition time and output loading sensitive delay model in delay estimation.

$$Temp_A = D_0 + \frac{T_r - T_1}{T_2 - T_1}(D_2 - D_0) \tag{3.2}$$

$$Temp_B = D_1 + \frac{T_r - T_1}{T_2 - T_1}(D_3 - D_1) \tag{3.3}$$

$$D_{target} = Temp_A + \frac{C_{target} - C_1}{C_2 - C_1}(Temp_B - Temp_A) \tag{3.4}$$

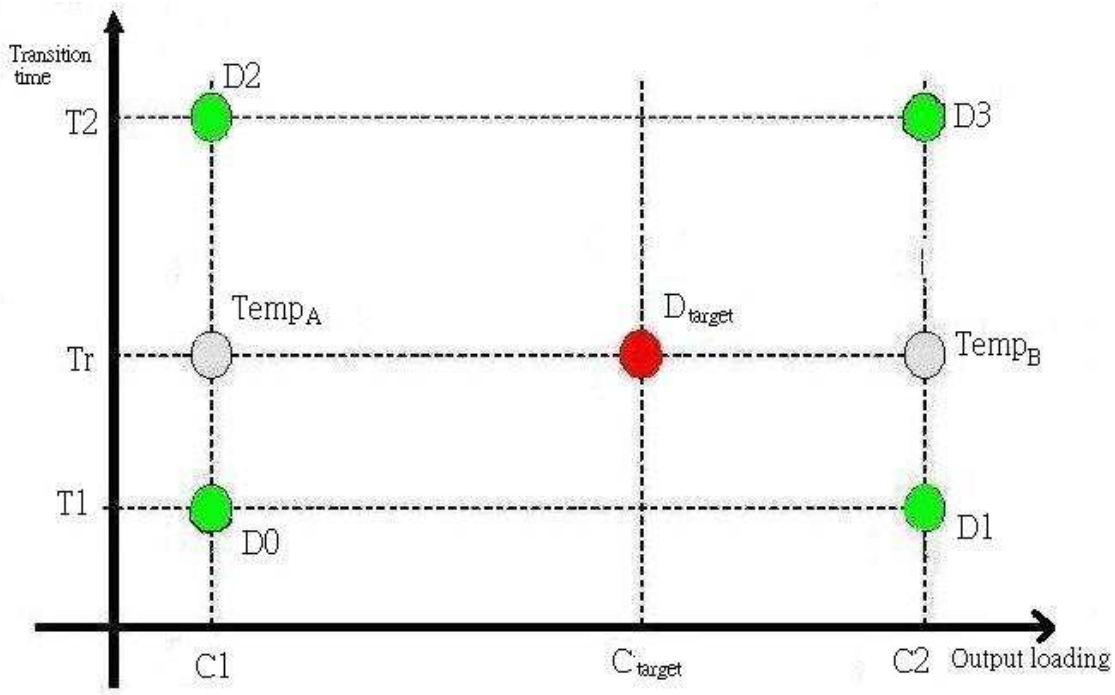


Figure 3.2: Interpolation of cell delay calculation.

3.1.2 Power Estimation

To optimize the power for clock tree, it is important to have a legitimate estimate of power consumption. In a clock tree, wires and cells contribute to the power consumption. For wires, the power $P_{sw}(wire)$ is dissipated by charging and discharging the wire capacitance. We use the given formula below to estimate the net capacitance for calculating the delay of clock buffers. The equation estimates the net capacitance for calculating the delay of clock buffers. The equation to estimate the net load C of a driver pin is

$$C_{wire} = \sum_{all\_fanout} net\_length * \phi \tag{3.5}$$

where ϕ is the weighting parameter from [1]. We can then obtain the wirelength by summing up the Manhattan distance of any two connecting cells ,which is the net length from driving cell to the driven cell.

The power consumption for cells consists of two components. One is switching power consumption $P_{sw}(cells)$ which corresponds to charging and discharging of the capacitance in cells. The other is an internal power (short-circuit power) of cells. We use table lookup based nonlinear power model library in this thesis to find accurate values of internal power for cells. The estimation of total power is then from the switching power (wire and cells) and internal power $P_{int}(cells)$ (short-circuit power) of cells:

$$P_{sw}(cells + wires) = (\sum_{i=1}^{all\_net} C_i) * V^2 * f \tag{3.6}$$

$$P_{total} = P_{sw}(cells) + P_{sw}(wires) + P_{int}(cells) \tag{3.7}$$

3.2 Problem Formulation

Problem 1 Low Power Buffered Clock Tree Synthesis: *Given a set of sinks (flip-flops) of the circuit as* $S=\{s_1,s_2,.....,s_n\}$ *and buffer library, construct a buffered clock tree which minimizes clock skew while reducing the power consumption on cells (flip-flops and clock buffers) and wires (wirelength).*

Chapter 4

Low Power Buffered Clock Tree Synthesis By Customized Buffer Insertion

Due to timing constraints, buffer insertion becomes necessary in clock tree synthesis. However, this technique may lead to more power penalty. Through effective analysis, we have made some observations for buffer insertion techniques in reducing clock power.

4.1 Find Best Transition Time Between Buffers and Flip-Flops

If we need to design clock tree for low power, we have to reduce wirelength that contributes switch power. Hence, for minimizing wirelength, we have to find a better topology in clock tree synthesis to reduce switch power in wires.

We can reduce the internal power consumption of cells by finding the relationship between power consumption of cells and transition time of the drivers. [13] and [14] propose the key point of transition time for reducing power consumption. They find that it is necessary to have a transition time bound to assure the performance of the system. This bound can affect the power optimization result significantly. In addition, in order to reduce the clock power, the internal power of cells need to be considered. We get two aspect of low power design. First, we need to carefully analyze the relationship between transition time and power consumption. Second, we have to consider the relations in the internal power of cells for total power consumption.

We discuss the first aspect from analyzing output loading and transition time between buffers and flip-flops. If we can find the best transition time, it can consume less power and insert less number of buffers in the circuit. Based on our observation, we can analyze the given buffer library and use the relationship between transition time of buffers and power consumption reduction. At the first level of clock tree, buffers need to drive some flip-flops. We observe that cells will consume power differently under different transition time of driving buffers.

From given buffer library, we understand the characteristics for buffers and flip-flops such as power, delay and transition time. We then analyze corresponding power for buffers driving flip-flops. We further know that under the best transition time, less power consumption can be achieved. For example, there is a circuit with 123 flip-flops and we choose one type of inverters in buffer library for repeater insertion. The simple correspondence relation is shown in Figure 4.1. In order not to insert many buffer to drive loads, we intend to use less number of buffers in clock tree

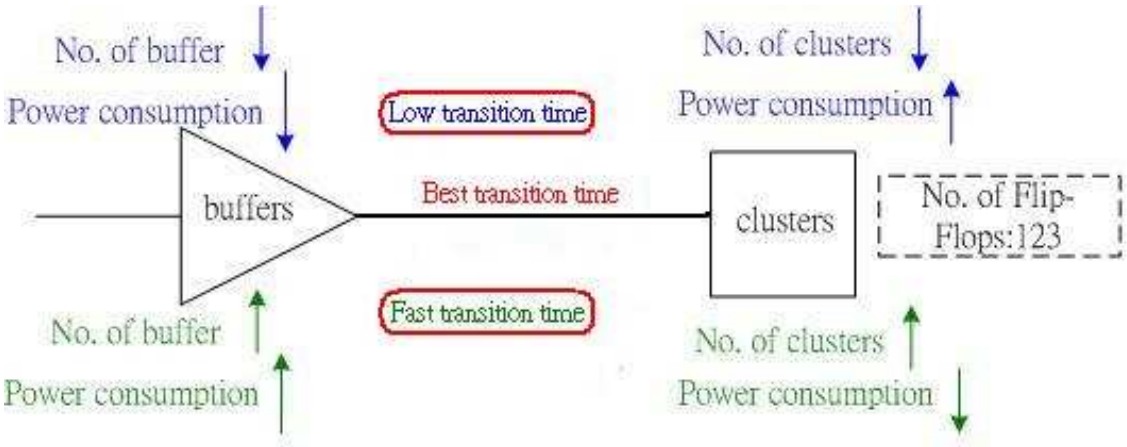


Figure 4.1: The simple correspondence relationship between choosing transition time for buffers and driver clusters of flip-flops.

construction, which means the number of load clusters will be less. However, buffers will provide larger transition time for flip-flops, introducing more power dissipated in flip-flop. On the other hand, if we have more clusters of loads, which means there are larger amount of buffers in the clock network. This leads to more power consumption in inserting buffers. According to our observation, we will analyze the buffer library to obtain the best transition time for the tradeoff. It is shown in Figure 4.2.

4.2 Use Smaller Buffer Type

We find the best transition time for each buffer in the library. Next step is to decide which buffer type will be used in clock design. Previous works emphasize that they can obtain optimal buffer size and insertion solution but they are not fitted

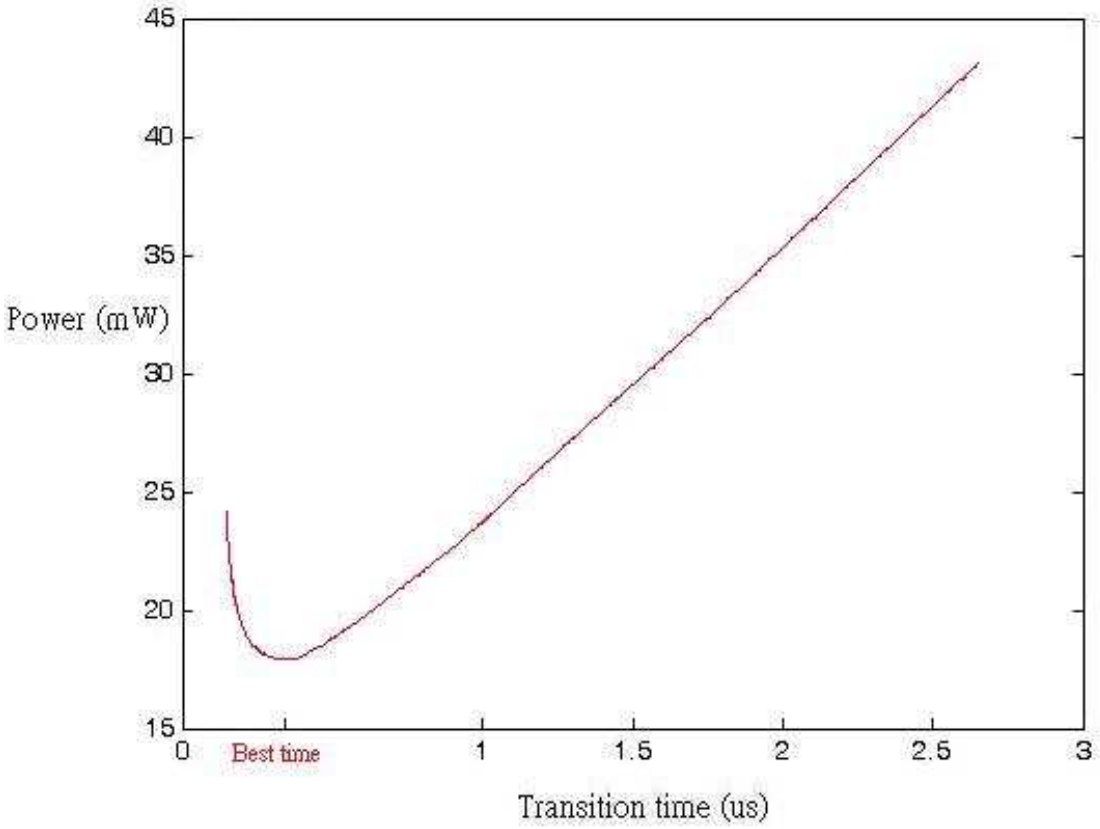


Figure 4.2: Different power dissipation in a cluster can be obtained due to different transition time between buffer and flip-flops.

into clock tree construction. Although we do not consider real routing, the switch power component for the net capacitance should be considered. We evaluate the wirelength by summing up the Manhattan distance of any two connecting cells. If we utilize the bigger buffer to drive some flip-flops, we will get the longer wirelengh from buffers to cells. If we choose smaller buffer type instead of the bigger one, we can obtain shorter total wirelength between buffers and cells.

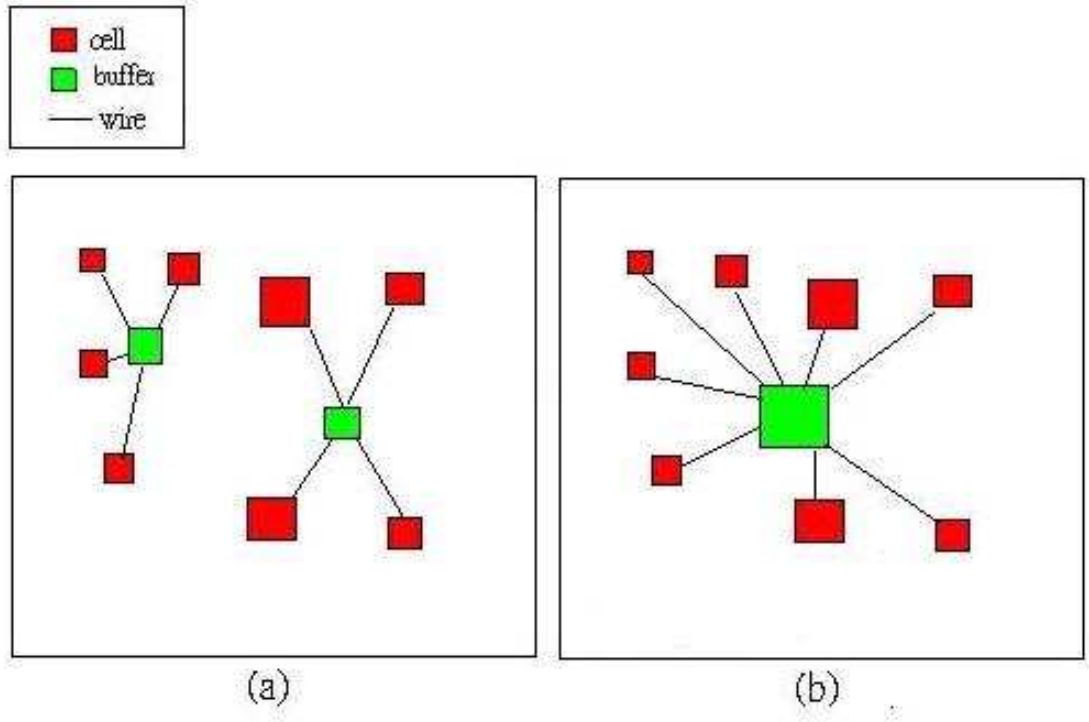


Figure 4.3: Consider two cases with different size of buffers. (a) We can reduce switching power component for wires by inserting smaller buffers. (b) Larger buffer will cause longer wirelength.

In Figure 4.3(a) and Figure 4.3(b) show different wirelength results. In Figure 4.3(a), we insert two small buffers and generate two clusters in the circuit and each buffer drives four flip-flops. however in Figure 4.3(b), we insert the buffer which has two times the size of small one, we utilize one buffer type to drive eight flip-flops. We observe that it has longer wirelength than using the smaller buffer sizes. Therefore we can reduce switching power component for wires by this observation.

4.3 Prefer Inverter When Inserting Clock Buffers

From the buffer library that includes clock buffers and inverters, we obtain each buffer's characteristic from previous analysis. Previous works seem not to use inverter as repeater since they do not mention there existing inverter type in buffer library. However we find that the power consumption of inverter is less than that of buffer type. Hence, we will consider to insert inverters as repeaters. Although we observe that using inverter type can reduce more power consumption, there exist a critical problem in using inverter insertion: phase problem. While all flip-flops are positive-edge-triggered, we need to ensure the clock signal that is positive-edge-triggered. It is not desired that the function of the circuit could be incorrect after clock tree construction. We solve the problem as shown in Figure 4.4. We need to keep an even number of inverters from the root of clock tree to leaves of the tree. Because we choose the same inverters to insert in each level of clock tree, this problem can be simplified. We just compute how many levels of clock tree are constructed, then decide to insert the buffer or the inverter in the root of clock tree. If we find the odd levels have been constructed, we need to choose the inverter type to insert in the root of clock tree, otherwise we will choose the buffer type to be inserted. This method can maintain the correction of phase problem and we can use effectively use inverter to reduce total power at the same time.

4.4 Clustering of Clock Tree for Load Balancing

There is a problem arises that how we can effectively utilize buffers we have analyzed. We can further create clusters for clock tree and let each cluster has approximately

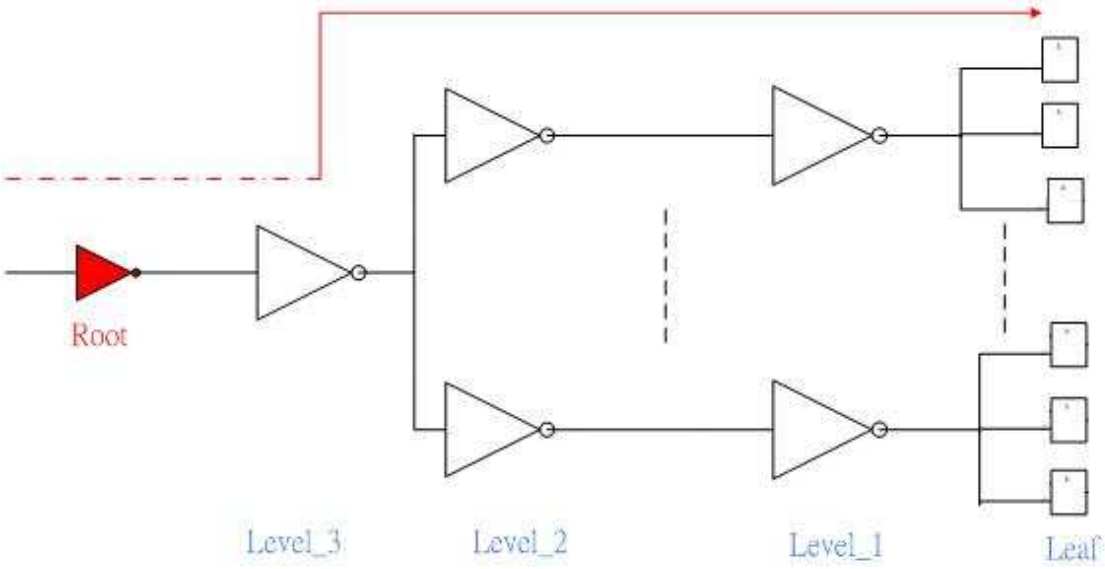


Figure 4.4: Keep an even number of inverters from the root of the clock tree to the leaf.

loading so that each buffer drive the same loading, which can ensure to maintain the best transition time in each cluster. We have the following problem: The clock pin clustering problem can be described as follows:
Given a set of clock pins (buffers), partition them into K clusters, and the ratio of the largest cost function to the smallest cost function is less than a given real bound B. Hence we have implemented a clustering algorithm, shown in Figure 4.5 , that can be used to create clusters for clock tree load balancing.

The task is to partition the input set into K clusters such that all of them have approximately the same load, then equally-sized buffers at the same level are inserted. This clustering can be applied hierarchically at different levels of a clock

Algorithm : Clustering (S,R,B)
Input :

1. A set S of N clock pins (buffers)
2. The rectangle R containing S
3. The balance bound B and each cluster size C

Output :

A list L of balanced clustering solutions

```
begin
    if set S = 0
      return L = {cost(S1), cost(S2), ...cost(Sk)};
    else
      compute clock pin set S1 contained by R
    end

    if |S − C| < B
      S = S − S1

    else
        do
          choose the nearest pins P of the rectangle bound ;
          if S − C > 0
            add P in S1
          else
            decrease P out S1
          end
        while |S − C| > B;
    end
end
```

Figure 4.5: The algorithm for clock pin clustering.

tree. At bottom level, the algorithm clusters flip-flops, while in middle levels of the clock tree clock buffers are clustered in the next level. The input of the algorithm is a set of flip-flops or clock buffers with associated locations and input capacitances, and the size of cluster loading. The total load of each cluster can be measured by a cost function:

$$C(each\_cluster) = \sum_{i=1}^{all\_cells} C_i + \beta * D \tag{4.1}$$

where C_i is the input capacitance of cells, β is the weight term, and D is the diameter of the input set which is defined as the Manhattan distance. Interconnect delays within clusters are concurrently balanced as well, thereby generating a low-skew buffered clock tree design. The synthesis approach can be applied recursively to generate a nearly zero skew clock tree. We show the result of clustering in Figure 4.6.

4.5 Overlap-Free Clock Buffer Placement

It is not feasible when inserted clock buffers overlap with other cells (shown in Figure 4.7(a)). Therefore we need to find the feasible and good range to place the clock buffers. First, if we find the clock buffers overlap with other cells, we define the feasible range around the clock buffer. If other cells are covered in the feasible range we will choose them to be candidates (shown in Figure 4.7(b)). Around these candidates we can find the feasible range according to the area of the clock buffer (shown in Figure 4.7(c)). We choose a feasible position around the cell in this feasible range, and then we place the buffer (shown in Figure 4.7(d)) in the feasible

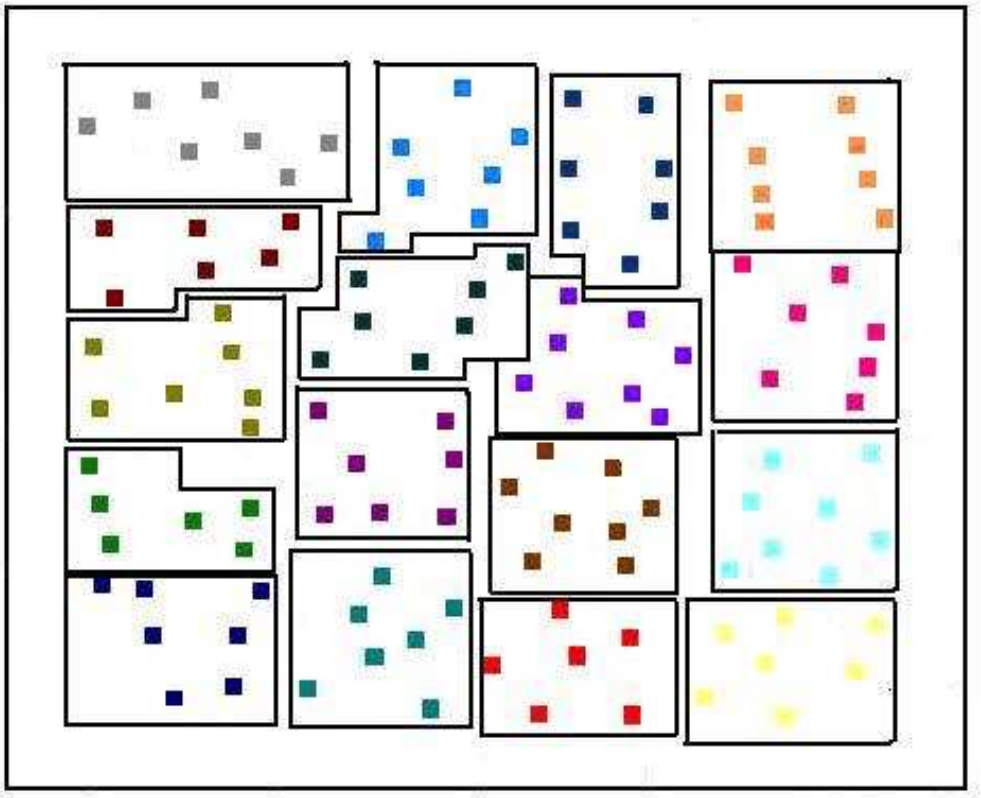


Figure 4.6: Flip-Flops are partitioned into clusters of approximately identical loading.

position. Figure 4.8 shows the overlap-free algorithm, which will avoid the clock buffer overlapping with other cells.

4.6 Methodology for Low Power Buffered Clock Tree Synthesis

We have introduced all of our observations for low power design and applied all of them in our methodology in achieving low power buffered clock tree construction. First we automatically analyze buffer library to obtain the characteristics of each

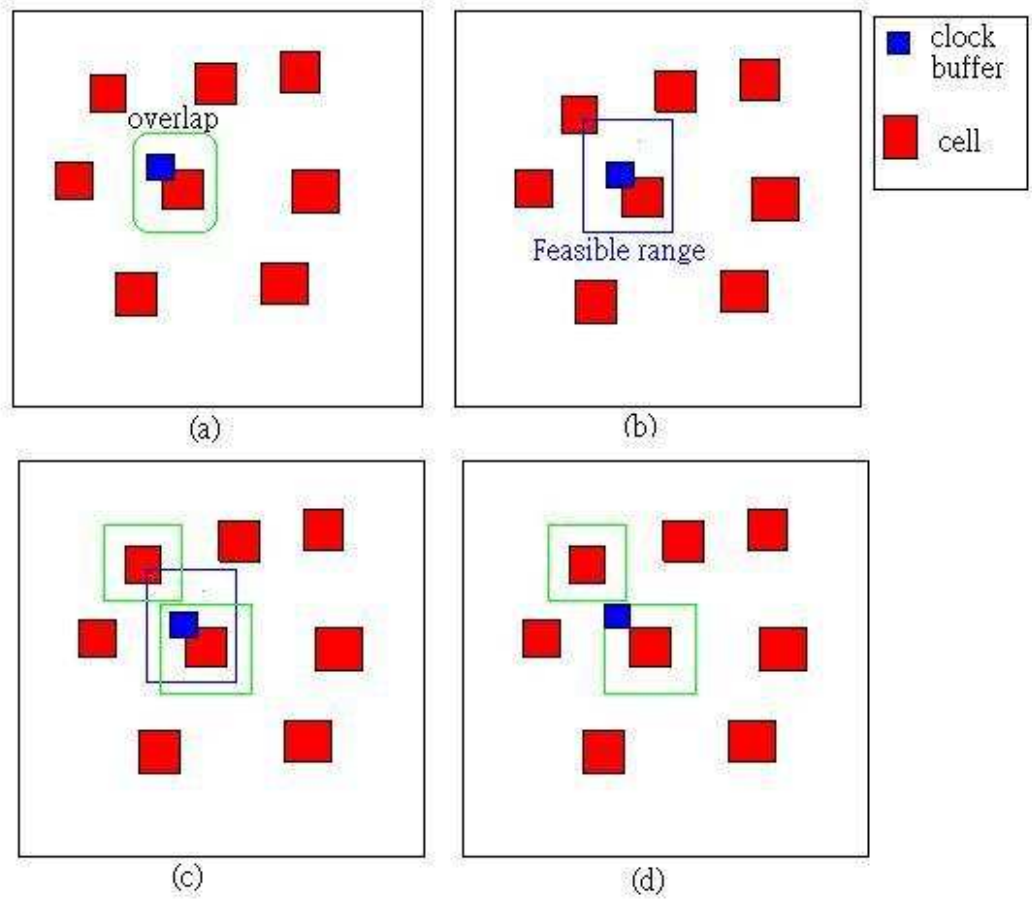


Figure 4.7: Overlap-free clock buffer placement.

buffer and find the best transition time for each buffer. We implement the clustering algorithm and consider the number of clusters which is based on the best loading in the circuit. Then we use the clustering algorithm in order to obtain each cluster of approximately equal capacitance loading and further insert buffers to drive identical loading at the same level of clock tree, also check if overlap occurs at the same time. Our design flow is shown in Figure 4.9.

```
Procedure Non-Overlapping Buffer Placement ()
begin
    while insert buffer in each cluster
        calculate the center of mean in clusters: mean_x, mean_y;
        overlap = check the overlapping(mean_x mean_y , buffer_area);
        if overlapping = 1//overlap occurs
            find the candidates of cells;
            find the feasible position: fea_x , fea_y
            return(fea_x, fea_y);
        else
            return(mean_x, mean_y);
        end
    end
end
```

Figure 4.8: The algorithm for avoiding overlapping during clock buffer placement.

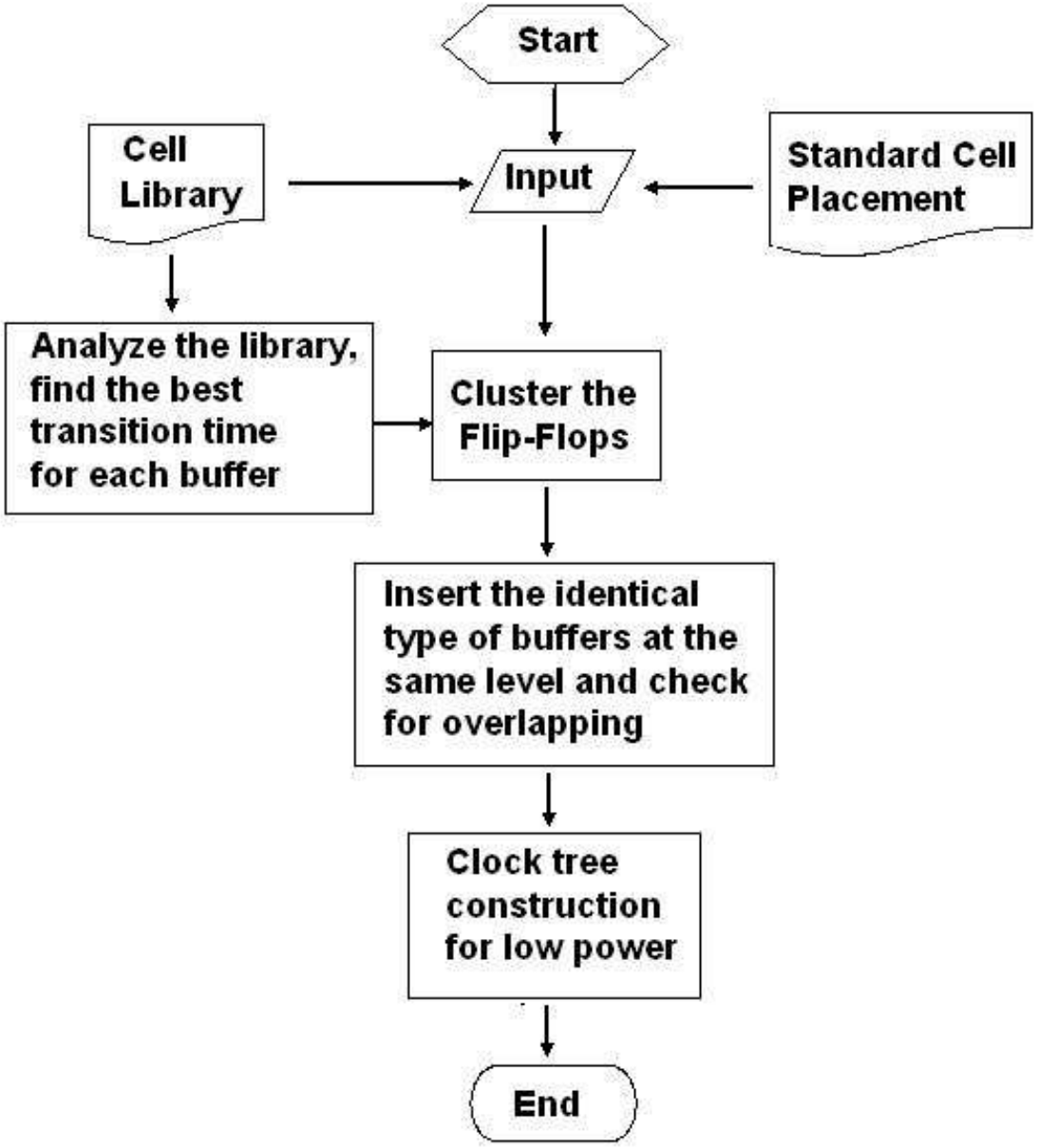


Figure 4.9: Design flow for low power buffer clock tree synthesis considering buffer transition time.

Chapter 5

Experimental Results

We have applied the design methodology shown in previous chapter to optimize the power consumption of the clock tree while conforming to the skew constraint and perform experiments on some real cell-based benchmarks provided in [1]. We also implement another combined approach [7] and [18] that is shown in Section 2.2 in order to show the effectiveness of our approach,. The parameters of technology for our experiments are form [1].

5.1 Gate Sizing vs. Our approach

In this section, we compared the gate sizing algorithm with the clock skew constraint on five benchmarks in Table 5.1.

We show the experimental results of power reduction ratio. Buffer library analysis finds the best transition time to optimize power dissipation, which means we make use of each buffer efficiently. Moreover, this approach can obtain more power reduction when the number of flip-flops is getting larger. The results show that our approach can achieve low power clock design and it is scalable to larger circuits. The

Gate sizing	Our approach
Benchmark	# of flip-flops
Clk_100	123
Clk_500	500
Clk_1000	1234
Clk_5000	5000
Clk_10000	10000

Table 5.1: Benchmark Profile from [1]

		GS	Ours	
Benchmark	Skew constraint(ns)	power(mW)		reduction(%)
Clk_100	0.05	4.48512	4.13609	8.4%
Clk_500	0.08	19.29948	17.51394	10.1%
Clk_1000	0.1	46.80549	42.30692	10.4%
Clk_5000	0.5	188.39265	169.13925	11.38%
Clk_10000	1	377.51768	331.92524	13.73%
Average				10.8%

Table 5.2: Comparison between gate sizing approach with our approach in power consumption.

results are shown in Table 5.2. Further-more, the approach in [7] may change the size of inserted buffers and that may cause the increase of clock skew. We, however, insert identical size of the buffers in the same level of clock tree, which effectively avoids the skew increase. It is shown significant improvement in clock skew in Table 5.3.

		GS	Ours	
Benchmark	Skew constraint(ns)	clock skew(ns)		reduction(%)
Clk_100	0.05	0.00849	0.00638	24.9%
Clk_500	0.08	0.03117	0.01867	40.1%
Clk_1000	0.1	0.05046	0.03454	31.5%
Clk_5000	0.5	0.1872	0.05682	69.6%
Clk_10000	1	0.21826	0.09648	55.8%
Average				44.4%

Table 5.3: Comparison between gate sizing approach and our approach in clock skew.

5.2 (N*N)Clustering + Gate Sizing vs. Our Approach

In this section, we add the (N*N) clustering algorithm [10] in gate sizing algorithm as a comparison platform. It will change the initial topology in the clock tree and can help the gate sizing algorithm to have more choices from the buffer library. We show the two experimental results in Table 5.4 and Table 5.5. In Table 5.4, the results of power consumption have been compared. It is shown that we have obtained up to 3.4% power saving averagely. In Table 5.5, we have smaller clock skew base on gate sizing as well.

5.3 Greedy Approach + Gate Sizing vs. Our Approach

In this section, we use the greedy based algorithm [18] in gate sizing algorithm as fairly comparison platform. It will generate the initial topology in the clock tree and insert buffer based on greedy approach in the clock tree. It will finally

		(n*n)clusters+GS	Ours	
Benchmark	Skew constraint(ns)	power(mW)		reduction(%)
Clk_100	0.05	4.20268	4.13609	1.5%
Clk_500	0.08	18.02451	17.51394	2.8%
Clk_1000	0.1	44.34551	42.30692	4.6%
Clk_5000	0.5	184.41550	169.13925	8.2%
Clk_10000	1	370.65985	331.92524	10.4%
Average				3.4%

Table 5.4: Comparison between our approach and gate sizing plus N*N cluster algorithm in power consumption.

		(N*N)clusters+GS	Ours	
Benchmark	Skew constraint(ns)	clock skew(ns)		reduction(%)
Clk_100	0.05	0.03197	0.00638	80%
Clk_500	0.08	0.06026	0.01867	69%
Clk_1000	0.1	0.09846	0.03454	64.9%
Clk_5000	0.5	0.20853	0.05682	72.7%
Clk_10000	1	0.38453	0.09648	75.9%
Average				72.5%

Table 5.5: Comparison between our approach and gate sizing plus N*N cluster algorithm in clock skew.

		Greedy+GS	Ours	
Benchmark	Skew constraint(ns)	power(mW)		reduction(%)
Clk_100	0.05	4.19938	4.13609	1.5%
Clk_500	0.08	17.96822	17.51394	2.5%
Clk_1000	0.1	43.62621	42.30692	3.0%
Clk_5000	0.5	176.7925	169.13925	4.3%
Clk_10000	1	347.2925	331.92524	4.4%
Average				3.14%

Table 5.6: Comparison between our approach and greedy based algorithm plus gate sizing algorithm in power consumption.

generate some good topology for the clock tree. However this approach does not consider transition time issue for low power. We show the two experimental results in Table 5.6 and Table 5.7. In Table 5.6, the results of power consumption have been compared for two approaches. It is shown that we have obtained up to 3.14% power saving averagely. However the results of clock skew (in Table 5.7) are slightly worse but we can still obtain near-zero clock skew. We analyze four approaches for power consumption in Figure 5.1. We find that if the numbers of flip-flops increase, our approach will save more power consumption and less clock skew is achieved in Figure 5.2. These verify the effectiveness of our approach.

		Greedy+GS	Ours	
Benchmark	Skew constraint(ns)	clock skew(ns)		reduction(%)
Clk_100	0.05	0.00601	0.00638	-6.0%
Clk_500	0.08	0.01767	0.01867	-5.6%
Clk_1000	0.1	0.03012	0.03454	-14.6%
Clk_5000	0.5	0.05248	0.05682	-8.2%
Clk_10000	1	0.09342	0.09648	-3.2%
Average				-7.52%

Table 5.7: Comparison between our approach and greedy based algorithm plus gate sizing algorithm in clock skew.

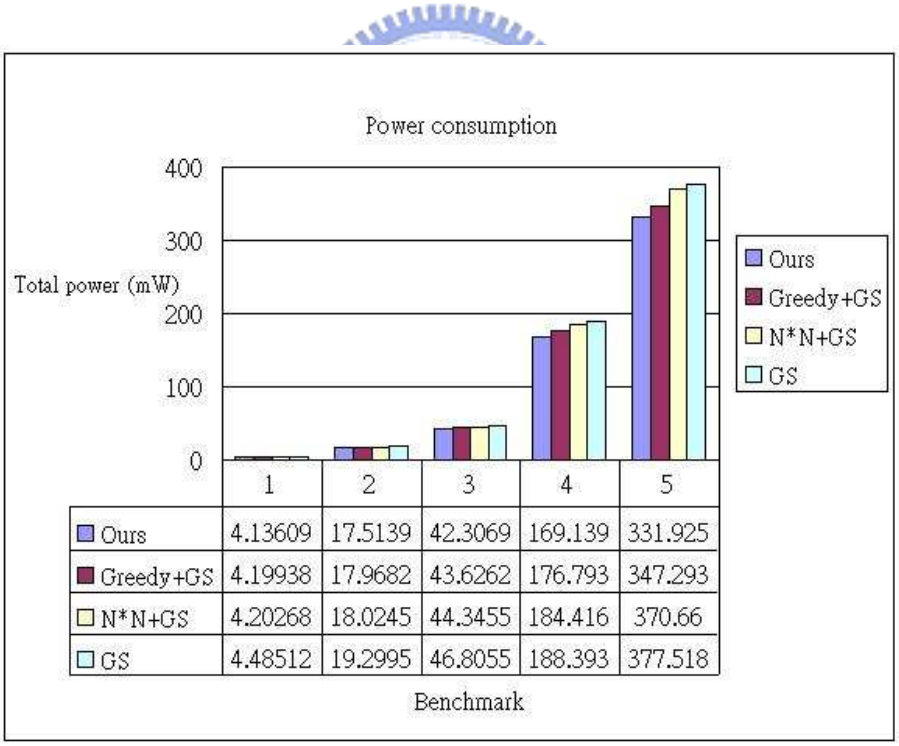
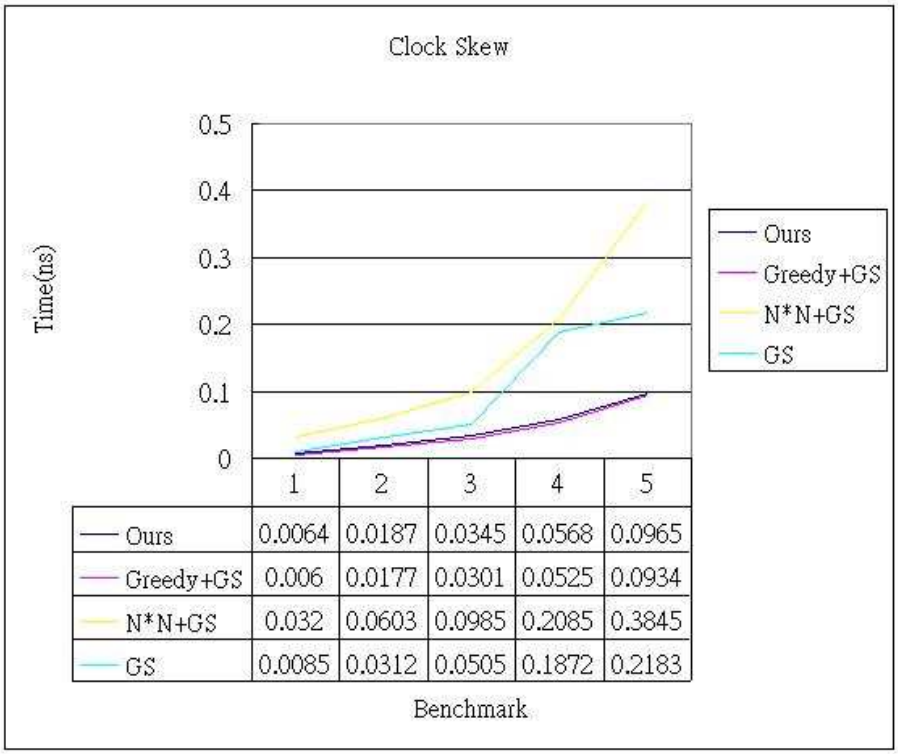


Figure 5.1: Clock power reduction comparison between approaches.



	1	2	3	4	5
Ours	0.0064	0.0187	0.0345	0.0568	0.0965
Greedy+GS	0.006	0.0177	0.0301	0.0525	0.0934
N*N+GS	0.032	0.0603	0.0985	0.2085	0.3845
GS	0.0085	0.0312	0.0505	0.1872	0.2183

Figure 5.2: Clock skew performance comparison between approaches.

Chapter 6

Conclusion and Future Work

A buffered clock tree is often desirable to solve skew problem. However, as technology advances, chips running at higher frequency are harder to obtain lower power consumption in clock network.

In this thesis, we further verify that the transition time is one of the key factors for low power clock design. We find that the transition time is more important for low power design because it affects the power consumption of the cells. We propose to analyze buffer library to find best transition time in buffered clock tree synthesis. According to the result of buffer library analysis, under the best transition time we attain the smaller power between the buffers and the flip-flops. We cluster flip-flops such that clusters are approximately the same loading. Each of these clusters is driven by identical buffer type. According to our analysis we ensure the power consumption of each cluster is the best solution and we can utilize the buffers more effectively. We have obtained averagely 10.8% power saving compared with a previous approach in aggressive gate sizing [7]. Due to the use of equally-sized buffer at the same level of clock tree, we can generate a nearly zero-skew clock tree

and it obtains lower power.

However, many researchers have discussed how to reduce power consumption considering the gate controlling. After the clock tree is constructed, the gating signal are optimizing for further power savings. Therefore we hope to find an effective algorithm that analyzes activation frequencies of flips-flops to effectively achieve less power consumption

Bibliography

[1] Taiwan IC/CAD Contest 2004. "http://www.cs.nthu.edu.tw/ cad/Problems.html".

[2] H. Bakoglu, J.T. Walker, and J.D. Meindl. "A Symmetric Clock Distribution Tree and Optimized High-Speed Interconnections for Reduced Clock Skew in VLSI and WSI Circuits,". In *IEEE International Conference on Computer Design*, pages 118–122, 1986.

[3] K.D. Boese and A.B. Kahng. "Zero-Skew Clock Routing Trees with Minimum Wirelength,". In *IEEE International Conference on ASIC*, pages 1.1.1–1.1.5, 1992.

[4] T.H. Chao, Y.C. Hsu, and J.M. Ho. "Zero Skew Clock Net Routing,". In *ACM/IEEE Design Automation Conference*, pages 518–523, 1992.

[5] Ting-Hai Chao and et al. "Zero skew clock routing with minimum wirelength.". In *Circuits and Systems: Analog and Digital Signal Processing, IEEE Transactions*, pages 799–814, 1992.

[6] Chunhong Chen and et al. "Activity-sensitive clock tree construction for low power.". In *Low Power Electronics and Design,*, pages 279–282, 2002.

[7] Olivier Coudert. "Gate Sizing for Constrained Delay/Power/Area Optimization.". In *Very Large Scale Integration (VLSI) Systems, IEEE Transactions*, pages 465–472, 1997.

[8] A.H. et al. Farrahi. "Activity-driven clock design.". In *Computer-Adied Design of Integrated Circuits and Systems,IEEE Transactions*, pages 705–714, 2001.

[9] J.P. Fishburn. "Clock Skew Optimization,". In *IEEE Trans. Computers*, pages 945–951, 1990.

[10] Prof. Shih-Hsu Huang. Personal Communication. The System VLSI/CAD Lab, Department of Electronic Engineering, Chung Yuan Christian University, Chungli, Taiwan.

[11] M.A.B. Jackson, A. Srivasan, and E.S. Kuh. "Clock Routing for Hight-Performance ICs,". In *ACM/IEEE Design Automation Conference*, pages 573–579, 1990.

[12] Ashish D. Mehta, Yao-Ping Chen, Noel Menezes, D.F. Wong, and Lawrence T. Pileggi. "Clustering and Load Balancing for Buffered Clock Tree Synthesis,". In *IEEE International Conference on Computer Design*, pages 217–223, 1997.

[13] Min Pan, Chris Chong-Nuen Chu, and J. Morris Chang. "Transition Time Bounded Low-power Clock Tree Construction.". In *Asia and South Pacific Design Automation Conference*, 2005.

[14] Min Pan, Chris Chong-Nuen Chu, and J. Morris Chang. "Transition Time Bounded Low-power Clock Tree Construction.". In *International Symposium on Circuits and Systems*, pages 2445–2449, 2005.

[15] S. Pullela, N. Menezes, and L.T. Pillage. "Low power IC clock tree design.". In *Custom Integrated Circuits Concerence*, pages 263–266, 1995.

[16] R.S. Tasy. "Exact Zero Skew,". In *IEEE International Conference on Computer-Aided Design*, pages 336–339, 1991.

[17] G.E. Tellez and M Sarrafzadeh. "Minimal buffer insertion in clock trees with skew and slew rate constraints.". In *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions*, pages 333–342, 1997.

[18] M. Vittal and A. Marek-Sadowska. "Low-power buffered clock tree design.". In *Computer-Adied Design of Integrated Circuits and Systems,IEEE Transactions*, pages 965–975, 1997.

VITA

Huang-Liang Chen was born in Taipei, Taiwan on July 30, 1980. He received the B.S. degree in Electronics Engineering from **N**ational Yunlin University of Science & Technology (NYUST) in June 2003 then entered the Institute of Electronics, National Chiao Tung University in September 2003. He specializes in the field of EDA. His advisor is Dr. Hung Ming Chen. He received the M.S. degree from National Chiao Tung University in June 2005.

E-mail: danny0730.ee92g@nctu.edu.tw