Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation for Designing Low Power Clock Network

Clock design plays an important role in modern VLSI designs. A chip may have millions of gates with a very complex structure. Studies have shown that the clock network contributes 20-50% of the total power on a chip. It is necessary to develop process to significantly reduce the power dissipation of the clock network, because of the growing importance of low-power designs for portable electronics. The design of the clock distribution network determines the clock skew, thus affecting the maximum attainable clock frequency. Clock tree design also determines the transition times of the signals at the clock elements, which again limits the maximum frequency of operation. Therefore the clock distribution needs more careful design planning methodology for modern VLSI.

The problem of clock tree synthesis for zero skew has been widely researched. The DME method [8] [6] optimally embeds a given clock tree topology in the Manhattan plane with zero-skew and attempts to minimize the total wire length. In addition to zero-skew, a second requirement on a clock tree is that the slew rate must be sharp. This requires the insertion of buffers to isolate the downstream capacitance. Various buffering algorithms [17] [15] have been proposed that can be used either

directly or adapted for clock routing.

High power dissipation is primarily due to the large amount of capacitance driven by the clock net. It already has many techniques to reduce the power consumption [16] [19]. Here, we propose a methodology for building low power buffered clock trees using a smaller voltage to distribute the signal over the chip, and then transmit the low-swing signal to low-swing flip-flops.

1.2 Organization of This Thesis

The rest of this thesis is organized as follows. Chapter II presents previous work, the delay and power estimation model and problem formulation. Chapter III introduces background of the proposed clock network and buffer insertion criterion. Chapter IV shows the proposed algorithm including clustering and DME with simultaneously buffer insertion. In Chapter V, we present the experimental results and Chapter VI concludes this thesis.

Chapter 2

Preliminaries

In this chapter, we give a brief overview of previous clock tree design. Moreover, we introduce the models that we use in this thesis, and formulate our problem as follows.

2.1 Previous Works

Previous works on clock tree construction concentrated on zero or near zero-skew routing. In addition to zero-skew, further work focus on routing clock tree with minimum total wire length and phase delay [8] [5] [7]. A bottom-up construction algorithm [14] reduced the total wire length by 15%, though its time complexity is $O(n^2 \log n)$. A clustering-based algorithm [4] improved the time complexity without any increase of the total wire length. The Deferred-Merge Embedding (DME) algorithm embeds internal nodes of topology G via a two-phase process. Let u and v be the nodes of a ZST (zero-skew tree). First, DME bottom-up merging scheme which ensures zero-skew under Elmore delay model is proposed and presents loci of possible locations of internal nodes in the zero-skew tree. A top-down phase then resolves the exact locations of all internal nodes in ZST.

The DME algorithm computes merging segment ms_v, representing the set of possible placement of v yielding a min-cost ZST rooted at v. The merging segment

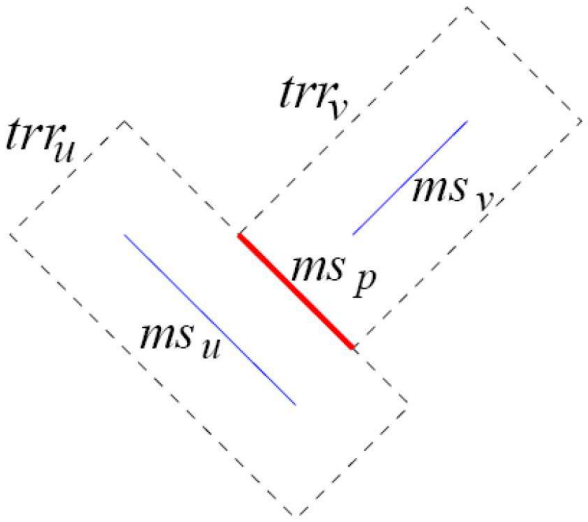


Figure 2.1: Construction of a merging segment $ms_p = trr_u \bigcap trr_v$.

is always a Manhattan arc, i.e., a segment having slope +1 or -1. Let p be the parent of u and v. The computation of ms_p is based on ms_u and ms_v. That is, ms_p is a set of points such that the ZSTs rooted at ms_u and the ZSTs rooted at ms_v can be merged with minimum added cost. For merging schemes, a widely accepted conclusion is that a subtree should be merged with its nearest neighboring subtree to save wire length.

The set of points within a fixed distance of a Manhattan arc is called a tilted rectangular region (TRR). The boundary of a TRR is composed of Manhattan arcs. The Manhattan arc at the center of a TRR is called its core. The radius of a TRR is the Manhattan distance between its core and its boundary. It has been shown that $ms_p = trr_u \bigcap trr_v$, where trr_u is the TRR with core ms_u and radius $|e_u|$ and trr_v is the TRR with core ms_v and radius $|e_v|$. It is shown in the Figure 2.1. During the second phase, DME algorithm determines exact locations for the internal node in a top-down fashion. [1] proposes a buffered clock tree synthesis applying a clustering algorithm to obtain clusters of approximately equal capacitance loading.

Since a clock network is normally very large, buffers are often employed to ensure

an acceptable slew rate. It is a constraint on the maximum allowable rise time on any buffer input signal. The constraint guarantees the correct operation of the clock tree and will be shown to be useful in manipulating the clock tree design. [10] places buffers of the same size at the nodes of the same level in the clock tree for two reasons: (1) zero skew routing generally results in a balanced tree; (2) this level by level buffering scheme can reduce the effect of inter-die process variations. However, low power design tends to have less number of buffers to reduce the power dissipation. [17] investigates the problem of computing a lower bound on the number of buffers required when given a maximum clock slew rate constraint and a predefined clock tree.

Clock networks account for a significant fraction of the power dissipation of a chip and are critical to performance. [16] presents theory and algorithms for building a low-power clock tree by distributing the clock signal at a lower voltage and translating the signal to a higher voltage at the utilization points. Two low-power schemes are used: reduced swing and multiple-supply voltages. Reduced swing clock scheme with drivers, intermediate drivers and receivers are illustrated in Figure 2.2.

Furthermore, the power dissipation of flip-flops can be decreased by some circuit techniques. Clustered Voltage Scaling(CVS), firstly proposed in [19], is a simple and practical technique for low power design. The essence of such technology is based on the utilization of excess timing slack in synchronous circuits. [13] shows that a low-swing clock double-edge triggered flip-flop(LSDFF) is developed to reduce power consumption significantly compared to conventional flip-flops. It uses a double-edge triggered operation as well as a low-swing clock. [12] also used the reduced swing clock driver and a special flip-flop which embodies the leak current cutoff mechanism.

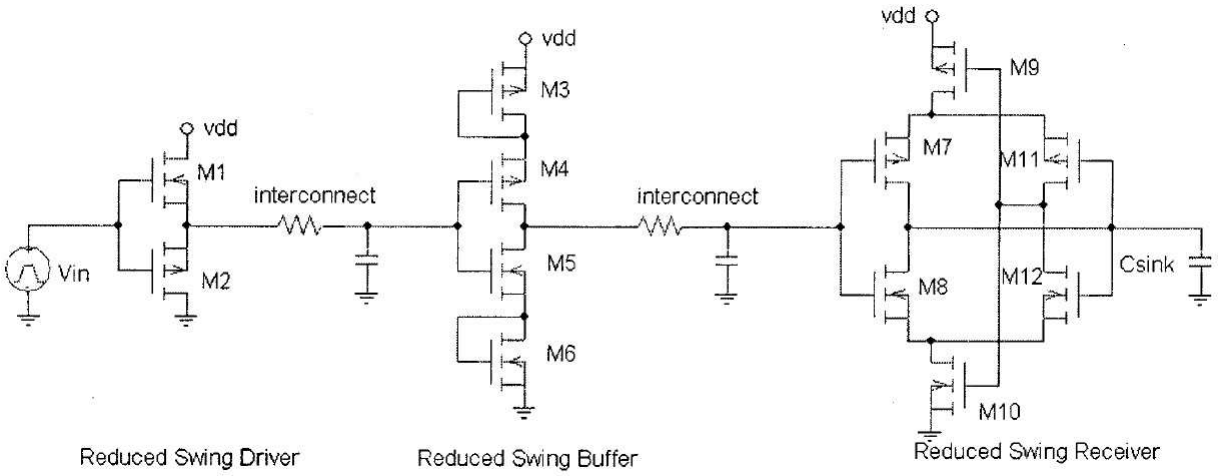


Figure 2.2: Reduced swing clock scheme [16], it is constructed by drivers, buffers and receivers.

2.2 Delay Computation and Skew Constraints

We assume that a clock tree is a binary tree in which the root is the clock source and the leaves are clock pins (flip-flops). Elmore delay model is used in computing the signal propagation delay of the clock tree. Let e_v denote the edge between an internal node v and its parent. Let r_v be the resistance of edge e_v and C_v be the total capacitance of subtree T_v. The delay from a node u of the clock tree to a descendant node v of u, denoted by *d(u, v)* is

$$d(u,v) = \sum_{u' \in path(u,v)} r_{u'} C_{u'}$$

Where path*(u,v)* is the set of nodes along the unique path from u to v excluding v. Let r be the root, N be the number of clock pins, and $li(1 \leq i \leq N)$ be a leaf. Clock skew can be defined as

$$S = \max_{0<i,j<N} \mid d(r,li) - d(r,lj) \mid$$

The model of the buffer is shown in Figure 2.3

In this model a buffer has three parameters: input capacitance c_b, output resistance r_b, and internal delay d_b. Buffers of different sizes have different parameters.

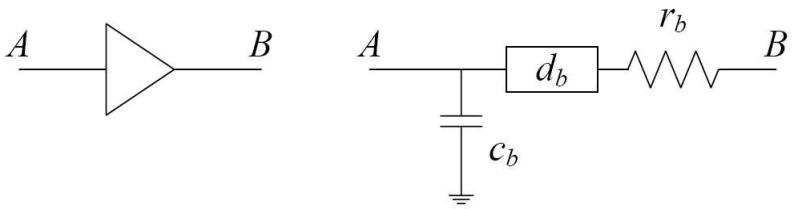


Figure 2.3: Buffer model has three parameters c_b, r_b, d_b.

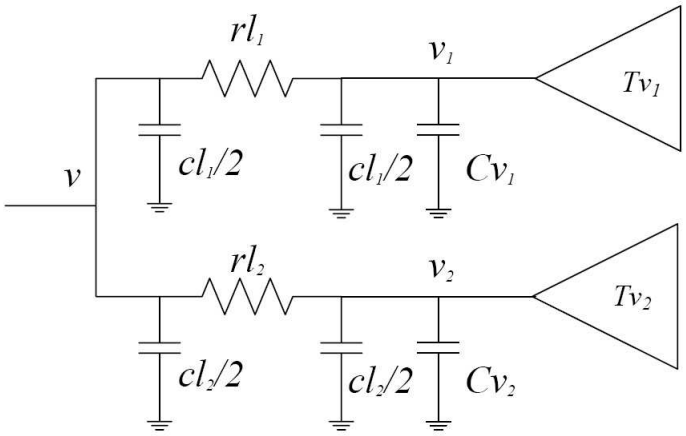


Figure 2.4: Zero-skew merging (with no buffers) v_1, v_2 are subtrees of parent node v.

If the tree topology is given, the position of the root v of the new subtree formed after merging can be determined from the positions of the two children v_1 and v_2 of v. This is illustrated in Figure 2.4 and 2.5. Let t_v denote the delay from v to any leaf of subtree T_v. Then

$$\begin{aligned} t_v &= rl_1(\frac{cl_1}{2} + C_{v1}) + t_{v1} \\ &= rl_2(\frac{cl_2}{2} + C_{v2}) + t_{v2} \end{aligned}$$

where l_1 *(l_2)* is the wire length from v to v_1 *(v_2)*, and r and c are per unit resistance and per unit capacitance of the routing wire. At a zero-skew merge, it is clear that $l_1+l_2 \geq l$, where l is the distance between v_1 and v_2. Therefore, in order to minimize the total wire length, it is desired to find v such that $l_1+l_2=l$. If there

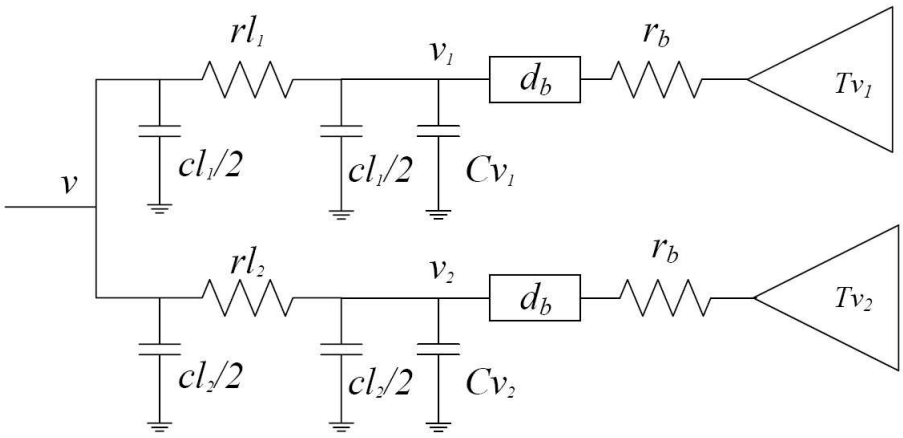


Figure 2.5: Zero-skew merging (with buffers inserted) v_1, v_2 are subtrees of parent node v.

is no such a v, zero-skew routing algorithms should use a "snaking". The number of snaking depends on algorithms and the input data. The same equations hold for the case in Figure 2.5, where node v_1 and v_2 are buffered. The relation between t_{vi} and t'_{vi} (i = 1, 2), the delay value before buffer insertion, is

$$t_{vi} = d_b + r_b * C_{vi} + t'_{vi}$$

During each merging distances l_1 and l_2 are determined such that the delay from the new root v to all leaves are all the same. In other words, zero skew is maintained.

2.3 Power Estimation of Clock Network

In modern VLSI designs, the high power dissipation in the clock net is primarily due to the large amount of capacitance driven by the clock net, in conjunction with the fact that the clock net switches on every transition. The total-power dissipation consists of two components: (1) the static-power dissipation, which is due to a leakage current of transistors during steady state. (2) the dynamic power dissipation, which has two components: short circuit and charge/discharge of capacitance power dissipation. The short circuit power dissipation is a function of the slew rate of the

input voltage; the sharper the clock edge, the lower the short circuit power dissipation.Moreover, the design trend is to use more pipeline stages for high throughput, which increases the number of flip-flops in a chip. Thus it is important to reduce power consumption in both the clock trees and the flip-flops. Power consumption of a particular clocking scheme ca be represented as

$$P_{ck-scheme} = P_{ck-network} + P_{FF}$$

where $P_{ck-network}$ and P_{FF} represent power consumptions in the clock network and flip-flops, respectively. Each term in (1) can be expressed as

$$P_{ck-network} = (C_{line} + C_{ck-tr}) * V_{ck-swing}^2 * f_{ck}$$

$$P_{FF} = (\alpha_i * C_i * \gamma + \alpha_o * C_o * \gamma + C_{ck-buf}) * V_{DD}^2 * f_{ck}$$

Where C_{line} is the interconnect line capacitance, C_{ck-tr} is the capacitance of the clocked transistors of the buffers, C_i is the internal node capacitance of the FF, C_{ck-buf} is the capacitance of the clock buffers inside the FF, C_o is the output node capacitance of the FF, $V_{ck-swing}$ is the clock swing voltage level,α_i is the internal node transition activity ratio,α_o is the output node transition activity ratio, and f_{ck} is the clock frequency. γ is 2 for double-edge triggered FFs and 1 for single-edge triggered FFs.

2.4 Problem Formulation

Same as other clock routing works, we adopt the Elmore delay model for delay computation. The total wire, buffer and flip-flop capacitance, resistance and low-swing clock voltage are considerations for the dynamic power consumption. The problem we will solve is formally stated follows.

Given a set of clock pins $S = \{v_1^s, v_2^s, ..., v_n^s\}$ *and a clock root* v_r*. A clock tree is defined by a tree rooted by* v_r *whose n clock pins are S(Figure 2.6). A set of pins in*

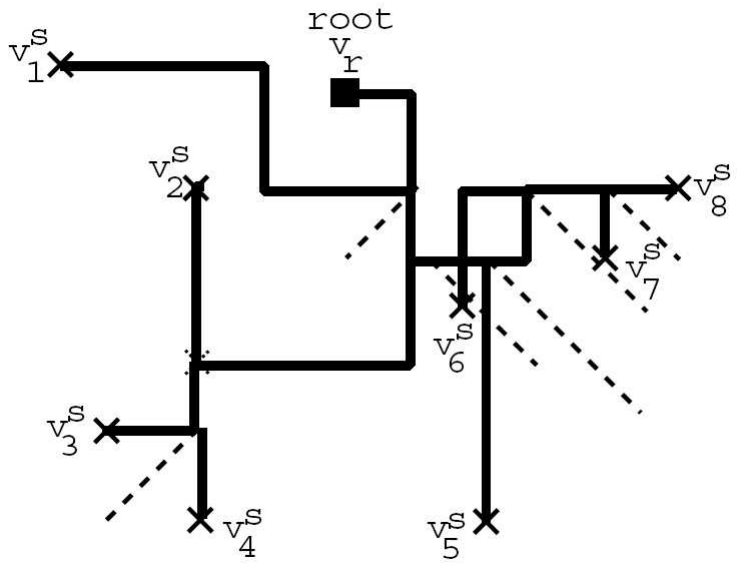


Figure 2.6: Clock tree routing with a root v_r and pins S=$\{v^s_1, v^s_2, ..., v^s_8\}$.

the subtree rooted by a node v is denoted by S_v. In this thesis, clock trees are always binary. We called the nearest internal node to the root denoted by v_c. We want to construct a zero-skew clock tree T rooted at V_r with clock sinks as leaf nodes such that the total power consumption is minimized. The transition time constraint is satisfied for any nodes for 10% of clock period.

Chapter 3

Low-Swing Buffered Clock Tree

Clock network plays an important role in power dissipation in a chip. In order to reduce power consumption, we use the low-swing clock voltage to translate the signal by low-swing buffers. Furthermore, LSDFF (Low-Swing Clock Double-Edge Triggered Flip-Flop) can use double-edge triggered operation as well as a low-swing clock. The details of the clock distribution are shown in the following subsections.

3.1 Structure of Low-Swing Clock Tree

In [16], multiple-supply voltage and low-swing interconnection schemes require additional LH converters that convert the incoming clock signal to the chip from a high-voltage swing to a lower-voltage swing. The input clock could be generated so that it has a low-voltage swing. The clock signal is then transmitted on the chip as a low-voltage signal, thereby ensuring a low power clock distribution network. We can save more power if we can transmit the low-swing signal directly to flip-flop without level converter. Therefore, we present a low-swing clock scheme with low-swing buffers and low-swing flip-flops. It is illustrated in Figure 3.1

The low-swing buffer is shown in Figure 3.2 from [12]. The clock swing, V_{clock} = V_{DD} - n*V_{th} depends on the number of inserted MOSFETs. The power consumption associated with the clock distribution is proportional to V_{clock} in this case. The

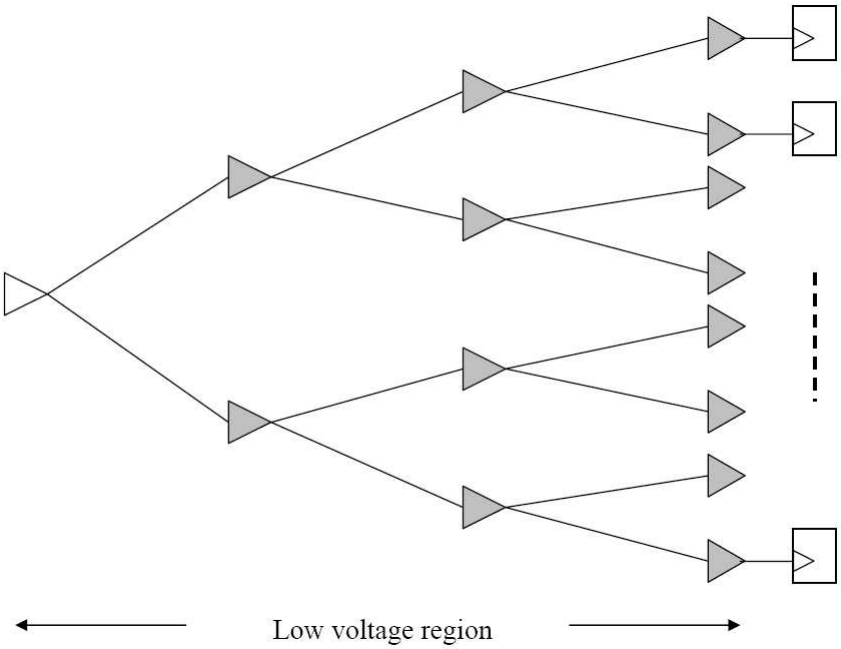


Figure 3.1: Low swing clock scheme with low-swing buffers and low-swing flip-flops.

buffers do not require additional supply voltage, so they are easily implemented. We simulate the low-swing buffer by SPICE under the condition of n=1, and V_{dd}=1.8V. The buffers can transmit the low-swing signal about 1.1V one by one during the simulation.

The low-swing flip-flop is presented in Figure 3.3 from [13]. It reduces power consumption in both the clock tree and the flip-flops. LSDFF uses both a low-swing clock and a double-edge triggered operation to reduce power consumption in the clock network. Further, LSDFF does not have any redundant internal data holding node switching. Therefore, we establish a SPICE model using 0.18um CMOS process for LSDFF. The simulation condition is 1.8V V_{dd} with clock frequency 125MHz. The low-swing clock voltage is about 1.1V, and the output load capacitance is approximately 100fF. The simulated waveform [13] of the LSDFF are shown in Figure 3.4 and 3.5.

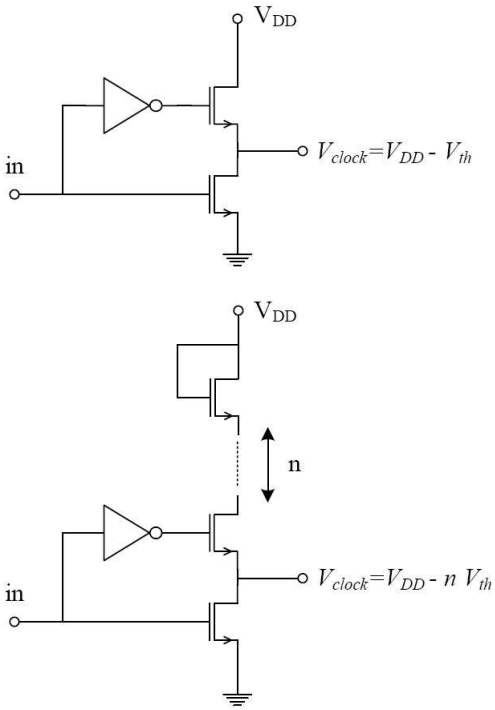


Figure 3.2: Low swing drivers $V_{clock} = V_{DD} - n*V_{th}$ [12].

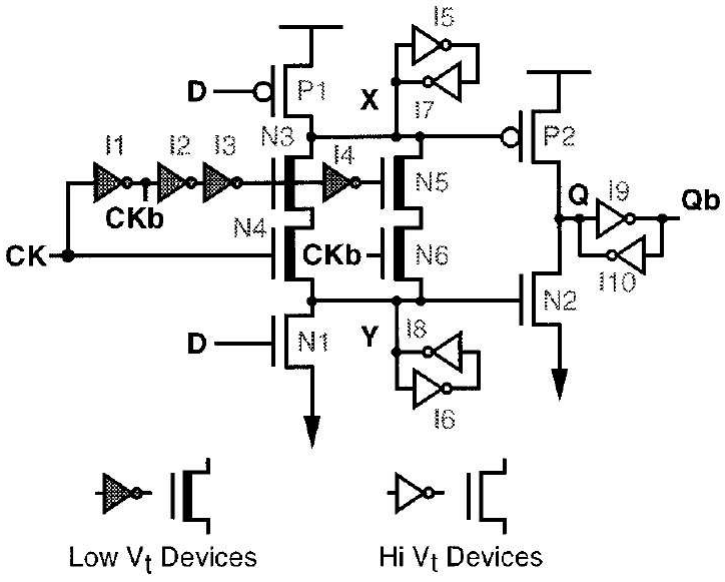


Figure 3.3: Scheme of LSDFF(Low-Swing Double-Edge Triggered Flip-Flop) [13].

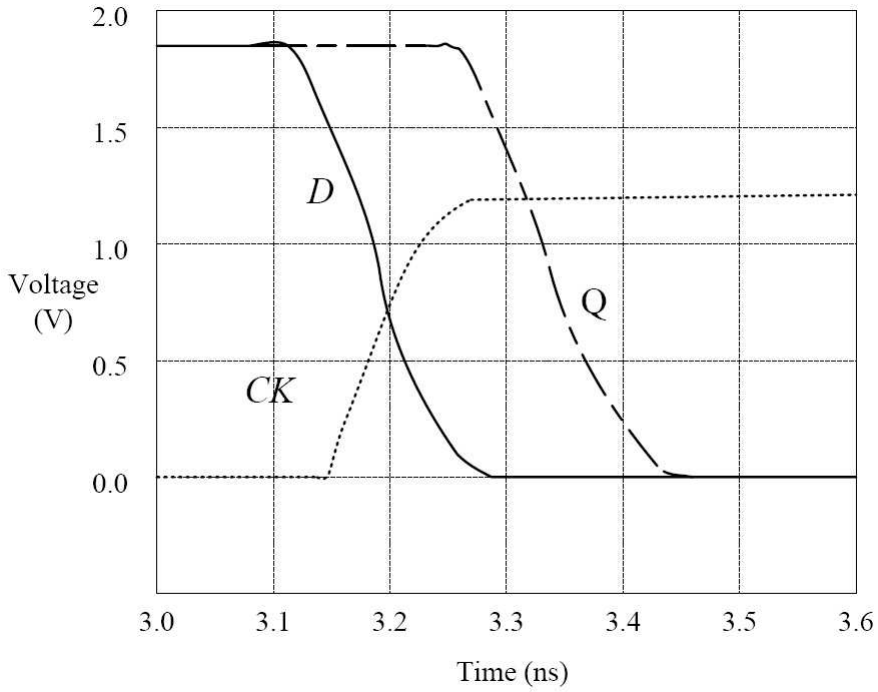


Figure 3.4: Simulated waveforms "High" to "Low" transition of Q at rising edge of the clock.

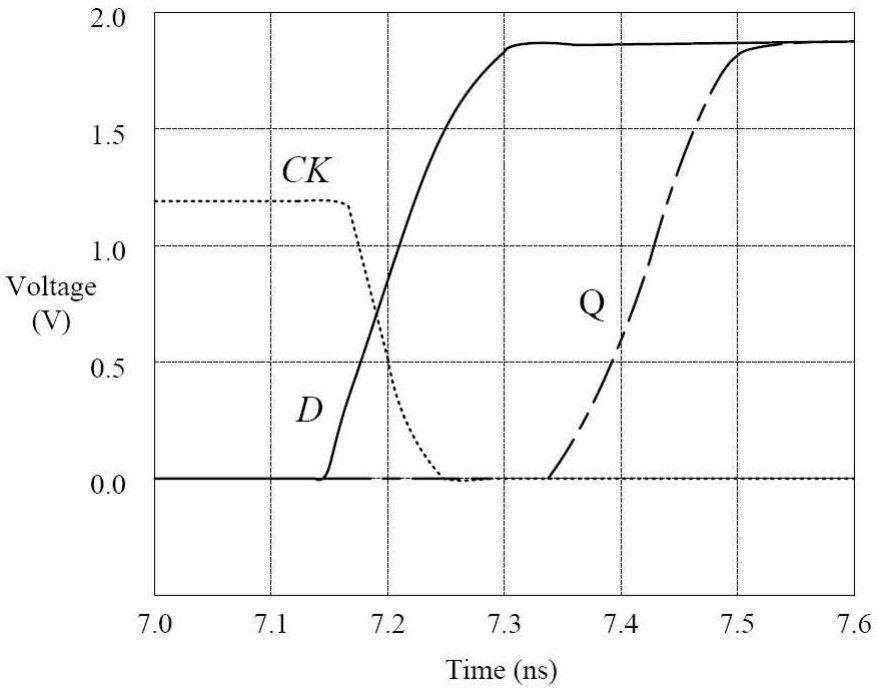


Figure 3.5: Simulated waveforms "Low" to "High" transition of Q at falling edge of the clock.

3.2 Buffer Insertion

A clock network distributes the clock signal from the clock source to the clock sinks. This must be done while maintaining the integrity of the signal, and minimizing the following clock parameter: (1) clock skew. (2) clock skew rate. (3) clock phase delay or latency. Consider a buffered clock tree, which is a tree that contains buffers in its source to sink paths. The addition of a buffer to a clock path reduces the wiring load and may also reduce the gate load to the clock driver. Hence, one may reduce the clock slew rates by adding buffers in a clock tree. The following facts motivate minimizing the number of buffers used in a clock tree: (1) total capacitance and the total power consumed in driving clock signal; (2) the number of buffers may be large enough to be of significant impact in total chip area.

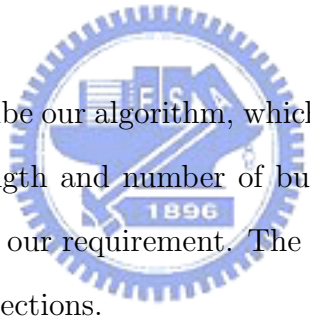
For this work, we make the following assumption. (1) the clock slew rate is a prediction of the 7% of maximum allowable transition time on any node input signal. This assumption guarantees the correct operation of the clock tree by checking the transition time violation during embedding phase. (2) the clock tree will be buffered with a single type of low-swing buffer. We employ SPICE simulation to construct a look-up table for our low-swing buffers. The variables of the table is input transition and load capacitance. The table gives out the output transition time. For the transition time after a wire, we employ the PERI (Probability distribution function Extension for Ramp Input) method [2] [15] to compute the output transition time. It is a simple technique for extending any delay metric derived for a step input into a delay metric for a ramp input for RC trees that is valid over all input slews. According to PERI method, the output transition time is the root-mean square of the output transition time of step input and input transition:

$$t_{out} = \sqrt{t_{in}^2 + t_{step}^2}$$

The output transition time of step input t_{step} is computed based on RC network charging process.

Chapter 4

Low-Swing DME Buffering for Low Power Clock Network Construction



In this chapter, we briefly describe our algorithm, which is based on DME algorithm. In order to reduce the wire length and number of buffers inserted, we modify the DME algorithm from [16] to fit our requirement. The details of these modifications are shown in the following subsections.

4.1 Load Clustering

The influence of clustering method on wire length are significant. The bottom-up phase during DME, the order of merging is very important. Furthermore, improper order of merging may cause the total wire length increasing and consume more power. For traditional matching-based methodology [11] [16], a geometric matching of *2N* segments or sinks cluster into N groups between segments, with no two of the N group sharing the same segments. This algorithm generates a good results due to the perfect matching when pins are regularly distributed. [6] proposed NS (Nearest-neighbor selection) algorithm. We modify the NS algorithm to adopt our methodology. Then, we introduce the NS algorithm.

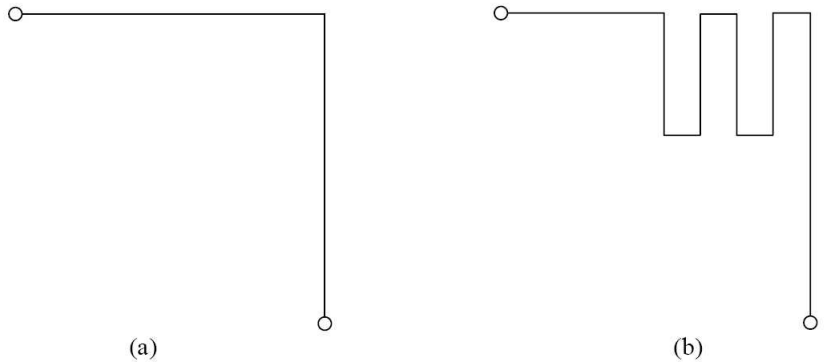


Figure 4.1: (a) A regular wire without wire snaking (b) Elongation of the wire by snaking [3] when unbalance merging occurs.

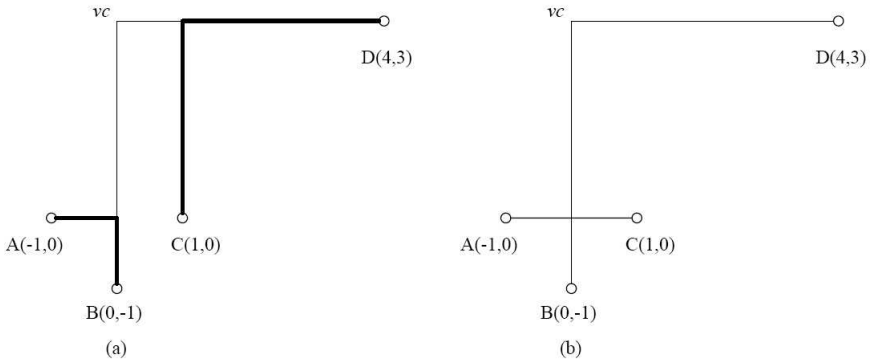


Figure 4.2: (a) Matching-based vs. (b) Ours clustering methodology, the routing by the matching method is longer than that by ours. The topologies are in Fig. 4.3.

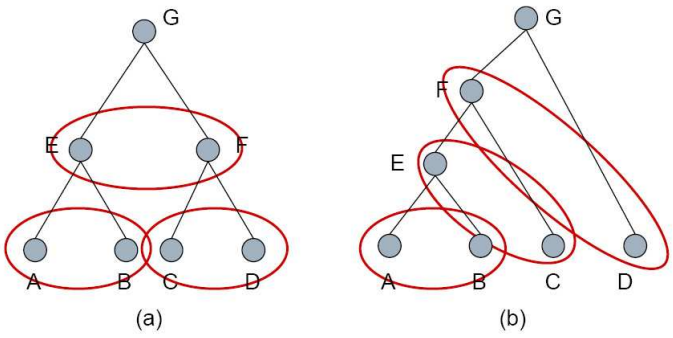


Figure 4.3: (a) Matching-based vs. (b) Ours clustering methodology when they build different topologies of clock tree with input {A, B, C, D}.

Let K be a set of points or diagonal segments with n elements. We take the nearest neighbor pair v_1 and v_2 from K, calculates a segment for v from v_1 and v_2 using the zero-skew merge, and put the segment into K, then, delete v_1, v_2 from K. After n-1 operation, K has only one element that is the segment for the center v_c. In Figure 4.2, the wire length by matching method is longer than that by ours methodology. It is clear that the topology of different methodology with matching-base and ours in Figure 4.3. [6] do not take buffer insertion into account. We categorize the nodes into buffered and nonbuffered node sets. NS algorithm is applied on nonbuffered node sets.

In case that the two merging segments are too much out of balance and the elongation severely affects the wirability, then addition of buffers and delay. A common practice for wire elongation is done by "snaking" Figure 4.1. It can be reduced if we choose a merging order that can reduce the delay differences among all merging segments. Because we do not have the topology of clock tree, we have to cluster the nodes (maybe merging sets or clock pins) to construct the binary clock tree. Merging cost is well-estimated by the following formulation:

$$Cost(i,j) = \frac{DISTANCE(i,j)}{dist_{half-perimeter}} + \frac{\mid Delay_i - Delay_j \mid}{delay_{half-perimeter}}$$

Cost(i,j) is the merging cost of the merging segment MS_i and merging segment MS_j. DISTANCE(i,j) is the Manhattan distance of MS_i and MS_j. $Delay_i$ means that the delay time from MS_i to the sinks. $dist_{half-perimeter}$ is the half perimeter of the chip core, and the $delay_{half-perimeter}$ is the RC delay of the half perimeter of the chip core $delay_{half-perimeter} = (r_0 * dist_{half-perimeter})(c_0 * dist_{half-perimeter})$. We normalize the distance and delay. Therefore, we can add these two factors to estimate the cost. The minimum cost pair has highest priority to be merged. By the consideration of delay and distance, we can avoid the merging of unbalance merging

Figure 4.4: Algorithm DME with buffer insertion

Algorithm Low-Power Clock Synthesis
Input: set of sinks S, timing constraints and technology parameters Output: tree of buffered merging segments with physical location

Begin
A = S; /* A is a set of non-buffered segments */
B = 0; /* B is a set of buffered segments */
While (|A| > 1 or |B| > 0)
{
if (|B| > 0 and |A| = 0)
A = B; B = 0;
while (|A| > 1)
{
choose the minimum cost of nodes b, c (segments or sinks);
a = *Zero-Merge* (b, c); /* zero merge the two segments */
A = A - {b, c}; /* delete merging segments b, c from A */
If (buffer insertion criterion satisfied /*if the transition
time exceeds the constraint */)
{
b = insert_buffer(b);
c = insert_buffer(c);
a = *Zero-Merge*(b, c);
B = B + {a};
} else
A = A + {a};
}
} /* bottom-up phase */
/* the last segment of A is for the center v_c */
search for nearest point to the root v_r on the segment v_c;
Local_Embedding(v_c);

segments.

4.2 DME with Buffer Insertion

4.2.1 Bottom-up phase

The clock tree construction procedure is similar to previous works [16] [1] [8] in many ways. The primary differences are in the use of a low-swing buffer for translate the low-swing clock voltage and low-swing flip-flops for clock sinks.

Figure 4.5: Subfunction of *Local_Embedding*

```
Local_Embedding(v)
{
  if ( v.lchild = NULL and v.rchild = NULL)
    return;
  let v1, v2 be the child of v, for i = 1, 2, determine a point vi on the
  segment for vi so as to satisfy the zero-skew.
  Route from v to vi;
  Calculate transition time of node v;
  Local_Embedding(v1);
  Local_Embedding(v2);
}
```

An outline of the algorithm is illustrated in Figure 4.4. The algorithm maintains two sets, A and B. A is a set of nonbuffered nodes and is initialized to a set of sinks, while B is a set of buffered merging segments and is initialized as an empty set. The merging segments correspond to the roots of subtrees and a buffered merging segment is the one that has buffers placed at the root of its two-child nodes.

A zero-skew clock tree is constructed using our algorithm. A simple flow chart of bottom-up phase is shown in Figure 4.6. It first clusters the point set (segment set) of A. We sort the cost of each pair of the merging segments. The minimum cost pair is selected from A, a zero-skew merging is performed for the merging segments that are selected by our clustering algorithm.

The two merged segments will be deleted from A and the new merging segment will be checked to see if it satisfies the transition time (slew rate) constraint. The transition time is calculated by PERI method. This may be verified by hypothetically inserting a buffer at the root of the new subtree and checking for transition time violations. In [16], it verifies the transition time violation by RC modeling of the merged nodes. The simulation results by RC modeling are not accurate because the input transition time takes an important role of transition time calculation.

We modified the methodology in [16] of the transition time calculation. The

following example shows the step of zero-merge and buffer insertion criterion. Let v_1 and v_2 are the minimum cost pair selected from A. *Zero-Merge()* is performed for v_1, v_2, and produce a new node v for a parent node of v_1 and v_2. DME bottom-up buffer insertion methodology can not predict the transition from the parent node v to the subtrees of child nodes v_1, v_2. From previous chapter, we hypothesize the input transition from root v equals 7% of the constraint. We will prove that our hypothesis is correct during top-down phase. The transition time of the child nodes v_1 and v_2 are the root-mean square of the transition time of parent node v and step input transition time. We have to check the transition time violation of v_1 and v_2. If such a violation exists (transition time is larger than 10% of the clock period), then buffers are inserted at the root of the two-child nodes and the two-child nodes are zero-skew merged again. Moreover, if buffers are inserted at the child nodes of a given node, then this node will be added to a buffered segment set B; if not, it is added to A. The procedure continues until the set A is empty, which occurs when all nodes at this level are buffered. Algorithm bottom-up buffer insertion ensures that for any path from the root to the sinks, there will be an equal number of buffers.

Once we have added the first level of clock buffers, we swap the sets A and B, and repeat the whole procedure again. The algorithm proceeds until there are only one node left in A and B is empty. At the end of the process, we has only one element that is the segment for the center v_c. At this point, the procedure returns a tree of segments rooted at v_c.

4.2.2 Top-down phase

Next, top-down embedding phase determines the best position for each node in the reverse order of bottom-up merging phase. First, the position of the center v_c is determined on the segment for v_c by routing from the root v_r in the shortest way. Once the position of a node is determined, we can easily calculate the positions of

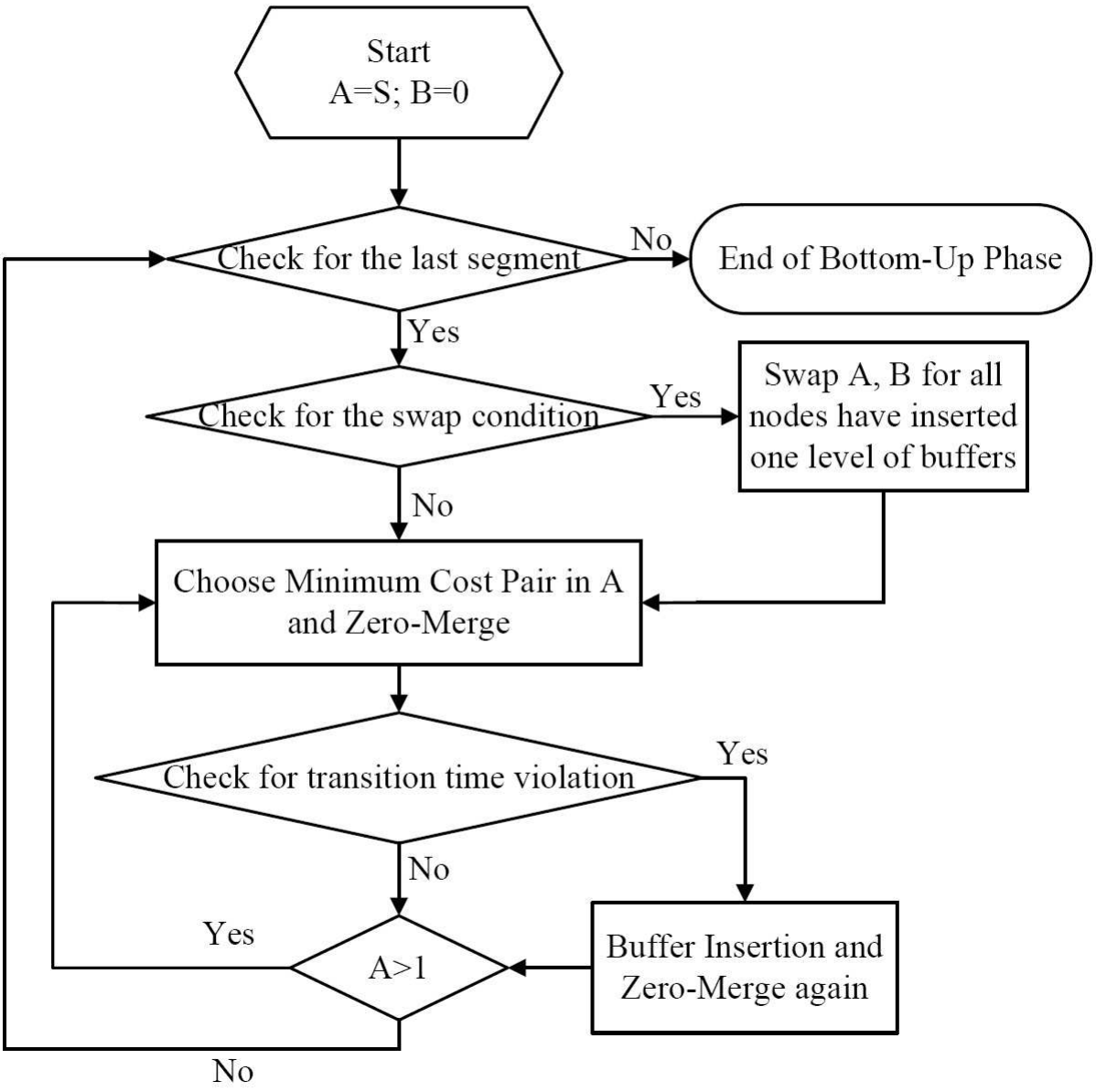


Figure 4.6: Process flow of DME buffering bottom-up phase.

Table 4.1: This table shows the relation of numbers of buffer inserted and hypothesis of input transition. Column "errors" means that the transition time violation for all nodes in clock tree during top-down phase. It verifies during top-down phase to re-calculate the transition time.

Hypothesis	6%		7%		8%	
	#buf	errors(%)	#buf	errors(%)	#buf	errors(%)
r1	23	1.4%	38	0.5%	40	0.3%
r2	51	1.2%	67	0.6%	84	0.4%
r3	66	2.1%	81	0.5%	112	0.2%
r4	87	1.8%	118	0.7%	223	0.2%
r5	174	2.0%	211	0.6%	314	0.3%

its children so as to satisfy the zero-skew merge equations. Therefore, all positions of nodes are calculated in a top-down fashion starting from the position of v_c.

In order to verify the transition time of each iteration, we calculate the transition time again by top-down fashion. We assume that the transition time of clock root is 7% of the clock period. By this assumption, we can obtain transition time results that are more accurate. We calculate the transition time downstream from the root and check for transition time violation. Table 4.1 shows the relation of numbers of buffer inserted and hypothesis of input transition. Column "errors" means that the transition time violation for all nodes in clock tree during top-down phase. If the transition time of a large part of nodes is under the transition time constraint, it means that our hypothesis is accurate. From experimental results, 7% is suitable for our hypothesis. It lowers the number of buffer inserted and controls the transition time violation less than 1% of total nodes.

Chapter 5

Experimental Results

We have implemented three approaches for low-power buffered clock tree in C++ and tested it on Pentium 4 PC 3.2 GHz with 1GB memory. One is the method from [10], and another approach is the low-swing clock distribution from [16]. Finally, it is our approach. In order to get more accurate simulation of power and timing analysis, we use UMC 180nm standard cell library for conventional buffers and FFs, and develop SPICE model for our low-swing FFs and buffers using 180nm CMOS process. The benchmark circuits are r1-r5 downloaded from the GSRC Bookshelf (http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/BST/). The parameters that we used are based on a 180 nm technology.

A comparison from Table 5.1 of the results of our algorithm with algorithm from [16] and [10] showed that our algorithm can reduce the total wire length. This is because that [16] use the bottom-up buffer insertion and does not have top-down phase to decide the physical position on the segment. Furthermore, our algorithm evaluates the cost with capacitance. By the consideration of capacitance, we can reduce the probability of of wire snaking by the capacitive load unbalance. According to the delay metrics used, the clock skew is zero by construction. In order to maintain the clock waveform, the transition time must satisfy the constraints accounting for 10% of the clock period. The three approach operate at different frequency. We set

Table 5.1: Performance Comparison (Phase delay, wire length and Power) between [16], [10] and ours. The results present advantages in wire length and number of buffers. However, our phase delay is average 6% more than [16]

	[16]			[10]			ours		
Benchmark	PD (ps)	WL (um)	#buf	PD (ps)	WL (um)	#buf	PD (ps)	WL (um)	#buf
r1	1281	191712	51	1013	193617	78	1330	192931	38
r2	1847	422109	89	1526	397432	125	2065	364872	67
r3	2337	532981	93	1939	479690	123	2413	453514	81
r4	2424	1038801	151	2205	962796	199	2572	890124	118
r5	2749	1589283	251	2571	1427090	350	2883	1380442	211

the transition time constraint equals *0.8ns*. We can see that the number of buffers can be reduced by our algorithm. However, our phase delay is average 6% more than [16].

Table 5.2 shows the comparison of power dissipation between algorithm [16], [10], and ours. The total power values shown in the table are the sum of the wire power, the buffer power and the flip-flops power. The fourth column results of my algorithm runs with frequency 125MHz with LSDFF. [16] and [10] runs with clock frequency 250MHz for conventional single-edge triggered FFs to achieve the same throughput. The low-swing voltage for LSDFF is about 1.1V. As shown in Table 5.2, the power consumption is reduced by our algorithm mainly due to halved clock frequency and low-swing voltage interconnection. It is reduced 48%-51% over [16].

Table 5.3 shows the performance comparison of different algorithm [16] and ours. In this experiment, We adopt the same structure of clock tree from Figure 2.2 [16]. The two simulation conditions use the same reduced swing drivers, buffers, receivers and FFs. The only different is the algorithm of constructing clock tree. The results present the slightly improvement of wire length and power consumption, though our phase delay is a little bit higher.

Table 5.2: Power dissipation of the clock tree between [16], [10] and ours. The results show that the power dissipation of our methodology is reduced 48%-51% over [16]

Benchmark	[16] for 250MHz (mW)	[10] for 250MHz (mW)	ours for 125MHz (mW)
r1	49.51	72.49	25.32
r2	103.20	153.53	64.90
r3	128.43	193.43	73.13
r4	273.54	396.60	141.30
r5	404.31	567.75	214.48

Table 5.3: Comparison of different algorithm([16] and ours) with the same structure of clock tree

	[16]			ours		
Benchmark	PD (ps)	WL (um)	Power (mW)	PD (ps)	WL (um)	Power (mW)
r1	1281	191712	49.51	1092	189201	45.81
r2	1847	422109	103.20	1931	365927	87.45
r3	2337	532981	128.43	2521	488218	106.21
r4	2424	1038801	273.54	2598	914891	242.84
r5	2749	1589283	404.31	2764	1383918	349.32

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 to obtain lower power consumption in clock network.

In this thesis, we have implemented our algorithm to generate a zero-skew clock tree. Our implementation guarantees that number of buffers along any path from root to sinks is equal, and uses a low-swing voltage to distribute the clock signal. Furthermore, we try to minimize the number of buffers under transition time constraints. By using the technique of low-swing interconnection (buffers and flip-flops), we obtain lower power consumption in clock tree construction. According to the results, we believe that our algorithm can be applied to most designs to construct a low power clock tree under appropriate constraints.

In future works, we plan to further improve our methodology to handle the gated clock designs. We can add the process of grouping the gated sinks, and construct the clock tree rooted by clock gates. After that, we move or merge for all the clock gates to optimize the power consumption and route one by one. We expect that the framework presented here can be extended in the presence of gated designs.

Bibliography

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

[2] Frank Liu Chandramouli V. Kashyap, Charles J. Alpert and Anirudh Devgan. "PERI: a technique for extending delay and slew metrics to ramp inputs". In *Proceedings of the 8th ACM/IEEE international workshop on Timing issues in the specification and synthesis of digital systems*, 2002.

[3] R. Chaturvedi and J. Hu. "Buffered Clock Tree for High Quality IC Design". In *In Proceedings of the IEEE International Symposium on Quality Electronic Design*, pages 381–386, 2004.

[4] Y. P. Chen and D. F. Wong. "An Algorithm for Zero-Skew Clock Tree Routing with Buffer Insertion". In *Proceedings of the 1996 European Design and Test Conference*, pages 66–71, 1996.

[5] Nan-Chi Chou and Chung-Kuan Cheng. "Wire Length And Delay Minimization In General Clock Net Routing". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 552–555, 1993.

[6] M. Edahiro. "A Clustering-Based Optimization Algorithm in Zero-Skew Routings". In *Proceedings IEEE/ACM Design Automation Conference*, pages 612–616, 1993.

[7] M. Edahiro. "Delay Minimization For Zero-Skew Routing". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 564–566, 1993.

[8] Ting-Hai Chao et al. "Zero Skew Clock Routing with Minimum Wirelength". In *IEEE Transaction on Analog and Digital Signal Processing*, pages 799–814, 1992.

[9] H.Zhang and J.Rabaey. "Low-Swing Interconnect Interface Circuits". In *Proceedings ACM International Symposium on Low Power Electronics and Design*, pages 161–166, 1998.

[10] A. Aziz I-M. Liu, T.-L. Chou and D. F.Wong. "Zero-Skew Clock Tree Construction by Simultaneous Routing, Wire Sizing and Buffer Insertion". In *Proceedings International Symposium on Physical Design*, pages 33–38, 2000.

[11] A. B. Kahng J. C. Cong and G.Robins. "Matching-Based Methods for High-Performance Clock Routing". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1157–1169, 1993.

[12] Hiroshi Kawaguchi and Takayasu Sakurai. "A Reduced Clock-Swing Flip-Flop (RCSFF) for 63Reduction". In *IEEE Journal of Solid-State Circuits*, 1998.

[13] Chulwoo Kim and Sung-Mo Kang. "A Low-Swing Clock Double-Edge Triggered Flip-Flop". In *IEEE Journal of Solid-State Circuits*, 2002.

[14] Y. M. Li and M. A. Jabri. "A Zero-Skew Clock Routing Scheme for VLSI Circuits". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 458–463, 1992.

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

[16] Jatuchai Pangjun and Sachin S. Sapatnekar. "Low-Power Clock Distribution Using Multiple Voltages and Reduced Swings". In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2002.

[17] Gustavo E. Tellez and Majid Sarrafzadeh. "Minimal Buffer Insertion in Clock Trees with Skew and Slew Rate Constraints". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 333–342, 1997.

[18] Ren-Song Tsay. "An Exact Zero-Skew Clock Routing Algorithm". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 242–249, 1993.

[19] Kimiyoshi Usami and Mark Horowitz. "Clustered Voltage Scaling Technique for Low-Power Design". In *Proceedings ACM International Symposium on Low Power Design*, pages 3–8, 1995.