Clock Period Constrained Minimal Buffer Insertion in Clock Trees

Gustavo E. Téllez and Majid Sarrafzadeh
Department of Electrical Engineering
and Computer Science
Northwestern University
Evanston, IL 60208
email: gus@cad2.eecs.nwu.edu

Abstract

In this paper we investigate the problem of computing a lower bound on the number of buffers required when given a maximum clock frequency and a predefined clock tree. Using generalized properties of published CMOS timing models, we formulate a novel non-linear and a simplified linear buffer insertion problem. We solve the later optimally with an $O(n)$ algorithm. The basic formulation and algorithm are extended to include a skew upper bound constraint. Using these algorithms we propose further algorithmic extensions that allow area and phase delay tradeoffs. Our results are verified using SPICE3e2 simulations with MCNC MOSIS 2.0μ models and parameters. Experiments show our buffer insertion algorithms can be used effectively for high-speed clock designs.

1 Introduction

Modern high-speed digital systems are designed with a target *clock period* (or *clock frequency*), which determines the system's rate of data processing. A clock network distributes the clock signal from the clock generator, or *source*, to the clock inputs of the synchronizing components, or *sinks*. This must be done while maintaining the integrity of the signal, and minimizing (or at least upper bounding) the following clock parameters:

- the *clock skew*, which is defined as the maximum difference of the delays from the clock source to the clock pins,
- the *clock phase delay* (or *latency*), which is defined as the maximum delay from the clock source to any clock pin,
- the *clock rise time* (or *slew rate*) of the signals at the clock pins, which is defined as the time it takes the waveform to change from a V_{LO} to a V_{HI} value.
- the sensitivity to parametric variations of the clock skew.

Additionally, these objectives must be attained while minimizing the use of system resources such as power and area. In high performance systems these constraints can be quite severe, consequently the layout of a good clock distribution network is difficult and time consuming. In this paper we will study some of these problems in buffered clock trees. Work on clock trees has focused on clock routing: in particular zero or near zero-skew routing, routing clock trees with zero-skew and minimal total wire length, and the construction of clock trees that minimize phase delay [CC93].

In the subsequent discussion, we will refer to the clock high and low times, which are the times during one clock period when the target clock signal is at logic '1' and logic '0', respectively. These times are respectively denoted T_{hi} and T_{lo}. In clock trees that use no buffers, the rise times of the clock signal must be smaller than the clock high/low times. It follows that the worse case rise time of the clock tree dictates a minimum clock period.

A *buffered clock tree* or *clock power up tree* contains buffers in its source to sink paths (see Figure 1). In this case the clock period is limited by the largest rise time of any buffer or sink input, and hence, it can be less than the latency of the network. Therefore, the largest rise time imposes a lower bound on the clock period. The previous discussion implies that a minimum number of clock buffers is needed to main-

© 1994 ACM 0-89791-690-5/94/0011/0219 $3.50

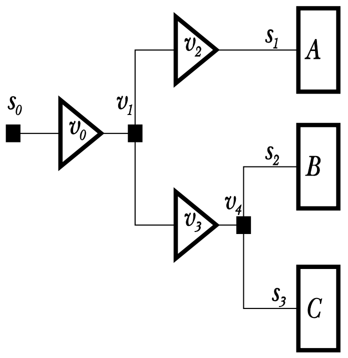

Figure 1: A buffered clock tree, with synchronizing elements A, B and C source s_0, sinks $\{s_1, s_2, s_3\}$, buffers $\{v_0, v_2, v_3\}$ and Steiner nodes $\{v_1, v_4\}$.

tain the clock signal integrity at a given clock period. The following facts motivate minimizing the number of buffers used in a clock tree:

1. The total capacitance, and therefore the total power consumed in driving the clock signal is minimized.

2. Impact on the total chip area is minimized.

In this paper we analyze the problem of computing the minimum number of buffers needed to satisfy a given clock period in an arbitrary clock tree. As far as we know, the problem of constructing a clock tree with minimal number of buffers, subject to a minimum clock period has not been posed. Work on buffered clock trees has focused on minimizing the phase delay of the clock tree [SW92]. More recently, work in [PMOP93] considers the minimization of skew and delay in the presence of process variations in buffered clock trees. For a survey on clock network construction issues see [Bak90, Fri93].

We make the following assumptions:

- A routing of the clock tree is given.

- The clock period constraint, which will be denoted $T_P = min(T_{lo}, T_{hi})$ is a constraint on the maximum allowable rise time on any buffer input signal.

- The clock tree will be buffered with a single type of CMOS buffer. This assumption is not unusual since using a single buffer type is an accepted strategy used in reducing skew and skew process-variation sensitivity.

We also consider several extensions to this problem:

- Compute the minimum number of buffers for a buffered clock tree with bounded *buffer skew*. The buffer skew is the largest difference in the number of buffers in any source to sink path. Compute a zero-buffer-skew solution which yields the minimum number of buffer levels.

- Compute the minimum number of buffers given the objective of minimizing the *longest buffered path* in a clock tree. The longest buffered path is the source to sink path with the largest number of buffers. This objective allows minimization of the clock phase delay.

- Given a library of buffers of different sizes obtain area and delay trade-offs. These trade-offs can also be obtained by varying the clock period constraint.

The outline of this paper is as follows. In Section 2 we introduce some definitions. In Section 3 we describe our assumptions about the timing models. In Section 4 we formulate the non-linear buffer insertion problem with no assumptions, and then proceed to apply the timing model assumptions to simplify the problem. We then formulate a linear buffer insertion problem, and in Section 5 we present an algorithm that solves this problem. Algorithm extensions are also discussed in Section 5. Finally, in Section 6 we present experimental results and conclusions.

2 Definitions and Terminology

A *clock net* N consists of a set of sinks $S = \{s_i \mid i = 1 \ldots n\}$, where $S \subset \mathcal{R}^2$, with a clock source node s_0. The sinks of the clock net represent the locations of the synchronizing components. A *clock tree* is a rooted tree $\mathcal{T}(V, E)$ over S with clock source s_0 being the root and S being the leaves of the tree. The nodes of the clock tree which are neither leaf nodes nor the root, are called *internal* nodes. An edge $e_{ij} \in E$ connects a parent node v_i and a child node v_j. We denote the number of children (or out-degree) of v_i as d_i. If node v_i lies in the path from node v_j to the root (leaf), then node v_i is said to lie above (below) node v_j. The length of the simple path from node v_i to a node v_j below it is given by $\lambda(i, j)$. A node v_i is said to be at level l if there are l edges on the path from v_i to the root. The height h of a tree is the largest level of that tree.

In a *buffered clock tree,* the internal nodes of the clock tree may represent buffers. Internal nodes which are not buffered are called *Steiner* nodes. A buffer tree section $\mathcal{T}_i(V_i, E_i)$, for a buffer $v_i \in V$, is a sub-tree of $\mathcal{T}$ rooted at v_i whose leaf nodes, or fan-outs of v_i, are either buffers or sinks of $\mathcal{T}$ below v_i. Internal nodes of the buffer section are Steiner nodes below v_i in $\mathcal{T}$.

A wire of length ℓ has wire capacitance $c = \beta\ell$. The input capacitance of the buffer or sink v_i will be obtained by $C_g(i)$. The total capacitance seen under a node v_i will be obtained from the function $C(i)$.

3 Buffer Timing Model Assumptions

We would like to obtain buffer insertion algorithms that do not depend on the use of a particular timing model, but rather, depend on the inherent delay properties of the buffers. With this idea in mind, we outline our assumptions about the buffer timing models. These assumptions will be stated referring to a single input, single output buffer:

- The input waveform of a buffer is parametrized by the rise time τ_r. The rise and fall time of the input waveform are assumed to be the same.

- The input rise time of a fan-out v_j in buffer section $\mathcal{T}_i$ can be computed using a function of the input rise time of buffer v_i and the wiring of the buffer section:

$$\tau_{r_j} = \Gamma(\tau_{r_i}, \mathcal{T}_i, j). \qquad (1)$$

We make two important assumptions about the function Γ, which we formalize in the next two properties:

1. **Property 1**: The function is strictly increasing.

2. **Property 2**: The structure of Γ implies that capacitive coupling is negligible.

Most CMOS timing models satisfy these properties [SK92] and the function can be implemented by any simulator, including SPICE.

4 Problem Formulation

In this section, we formulate the so-called *non-linear constrained buffer insertion problem.* A rooted clock tree $\mathcal{T}$ is given with an assigned input rise time τ_{r_o}. The *stage rise time* of a buffer stage $\mathcal{T}_i$ is defined as the largest rise time at a fan-out and is denoted by τ_{r_i}.

Non-Linear Constrained Buffer Insertion Problem (NLCBI): Minimize the number of stages $\mathcal{B}$ in a given rooted clock tree $\mathcal{T}$, such that for every stage $\mathcal{T}_i$, $i = 1, \ldots, \mathcal{B}$, the stage rise time does not exceed the clock period: $\tau_{r_i} \leq T_P$.

The NLCBI formulation is difficult to manipulate, so we will first consider a related problem (**Problem I**): let the tree $\mathcal{T}$ consist of a chain of $\mathcal{B}$ buffers such that $\lambda(i-1, i) = \ell_i$. **Problem I** is to maximize $\mathcal{L} = \sum \ell_i$. We solve to **Problem I** using the following theorem:

Theorem 4.1 *: Finding ℓ_i, $i = 1, \ldots, \mathcal{B}$, such that:*

$$\tau_{r_i} = \Gamma(\tau_{r_{i-1}}, \mathcal{T}_i, \ell_i) = T_P \qquad (2)$$

is an optimal solution to **Problem I**.

Theorem 4.2 *: The NLCBI problem is separable and increasing.*

Problem I is solved by computing ℓ_i from Equation (2) at each stage $i = 1, \ldots, \mathcal{B}$. This is done by using a standard line search technique and the solution of the previous stage. Notice that Equation (2) forms a single variable (ℓ_i) non-linear difference equation. We have investigated the properties of this equation with two experiments which we describe next.

In **Experiment E1** we iterate placing buffers in a chain such that the stage rise time satisfies $\tau_{r_i} = T_P$. We evaluate τ_{r_i} only for one stage at a time. We stop when ℓ_i converges. In our experiments we use SPICE3e2 as the rise time evaluation function. To show the accuracy of this approach, we propose **Experiment E2,** where each evaluation is done to the complete buffer chain of i buffers. We have run several cases of these experiments and in all cases ℓ_i

Rise	ℓ_∞			
Time	2x4		2x30	
T_P (ns)	**E1**	**E2**	**E1**	**E2**
5	2502	2330	9042	8836
10	8479	8452	22589	22507
100	76754	76765	105549	105573
500	218068	218475	249104	249559

Table 1: Values for ℓ_∞ obtained for different buffers and a range of T_P. **E1** and **E2** show close agreement.

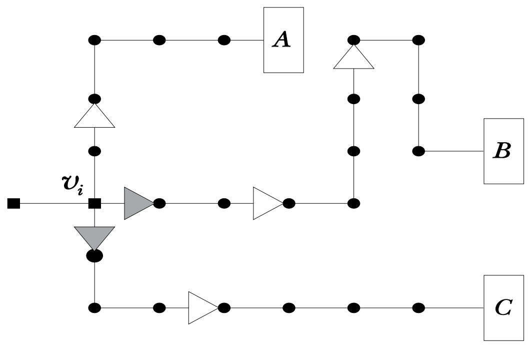

Figure 2: Example for buffer insertion with $\ell_\infty = 4$ and $\epsilon = 1$. after **packNode** on v_i. Buffers inserted by **packNode** are highlighted.

has converged. Furthermore, experiments **E1** and **E2** have shown excellent agreement as shown in Table 1.

We use the wire length at this last stage of the experiment to define two quantities: $\ell_\infty = \ell_i + C_g(i)/\beta$ and $C_\infty = \beta\ell_i + C_g(i)$. From our experimental evidence we conjecture that any buffer stage k with $C(k) \leq C_\infty$ will have $\tau_{r_k} \leq T_P$. We argue that we can use C_∞ as an upper bound on the capacitance of any buffer stage in the tree. We can now formalize the linear constrained buffer insertion problem:

Linear Constrained Buffer Insertion Problem (LCBI): Minimize the number of stages $\mathcal{B}$ in a given rooted clock tree $\mathcal{T}$, such that for every stage $i = 1, \ldots, \mathcal{B}$, the stage capacitance does not exceed the upper bound capacitance: $C(i) \leq C_\infty$.

5 An Exact Algorithm

In this section we propose an algorithm, so-called **buBufferInsert**, that solves the LCBI problem as follows: use **E1** or **E2** to compute ℓ_∞, then traverse the tree in bottom-up order. Buffers are inserted at two stages of the tree traversal, each of which is modelled as a packing problem. The algorithm **packEdge** packs a given edge e_{ij} with a chain of buffers, at a distance of ℓ_∞ from each other. The **packNode** algorithm adds buffers at a node v_i after **packEdge** has visited all its fan-out edges. The algorithm first selects a maximum number of fan-out edges that will require no additional buffers. A single buffer is inserted in the remaining fan-out edges. Each buffer v_k inserted by **packNode** has $\lambda(i,k) = \epsilon$, which represents a minimum buffer spacing. We illustrate these algorithms in Figure 2. Upon completion, **buBufferInsert** returns the number of buffers inserted, $\mathcal{B}$ and as a by-product, has inserted these buffers into the proper locations in the clock tree.

The total work done by the algorithm is $O(\mathcal{B} + \sum_{i=1}^{n-1} d_i \log d_i)$ (the **packNode** algorithm sorts the fan-out edges of each node). For trees of interest, the maximum out-degree $d_{max} = O(1)$, hence the time complexity is $O(\mathcal{B}+n)$. Since the tree and buffer data are stored as part of a tree data structure, the space complexity of this algorithm is also $O(\mathcal{B}+n)$. A modified method reduces the time and space complexity of the algorithm to $O(n)$.

The **buBufferInsert** algorithm unfortunately produces solutions of arbitrary buffer skew. Skew can have several causes:

- The number of buffers is not the same between two paths,
- The loads in the buffer stages are not matched properly.

We only consider the first source of skew. The other sources of skew can be handled using the methods proposed in [PMOP93]. The length of the longest and shortest buffered paths in a tree rooted at node v_i, are denoted $lp(i)$ and $sp(i)$, respectively. The buffer skew at a node v_i is defined as $Skew(i) = lp(i) - sp(i)$. We propose a simple modification to **buBufferInsert** such that $Skew(0) \leq \mathcal{S}$. The formulation of this problem is as follows:

Linear Constrained Buffer Insertion Problem with Bounded Skew (LCBI-BS): Minimize the number of stages $\mathcal{B}$ in a given rooted clock tree $\mathcal{T}$, such that for every stage $i = 1, \ldots, \mathcal{B}$, the stage capacitance does not exceed the upper bound capacitance: $C(i) \leq C_\infty$ and the buffer skew satisfies $Skew(i) \leq \mathcal{S}$.

The **LCBI-BS** problem is solved by modifying the **packNode** step in the **buBufferInsert** algorithm. The modified algorithms are called **packNode-BS** and **buBufferInsert-BS**, respectively. The modified algorithm will first insert buffers so that $\mathcal{S}$ is satisfied, i.e. $lp(i) - sp(i) \leq \mathcal{S}$. Next, if necessary, the **packNode** algorithm is applied to those fan-outs whose skew is less than $\mathcal{S}$. Finally, if necessary, a layer of buffers is added to the fan-outs with skew $\mathcal{S}$.

We have experimentally found that our buffer insertion algorithm decreases the longest buffered path while it reduces the buffer skew. We conjecture that, for a given tree, if two solutions from the **buBufferInsert-BS** algorithm, A and B have skews $\mathcal{S}_A < \mathcal{S}_B$, then $lp_A \leq lp_B$.

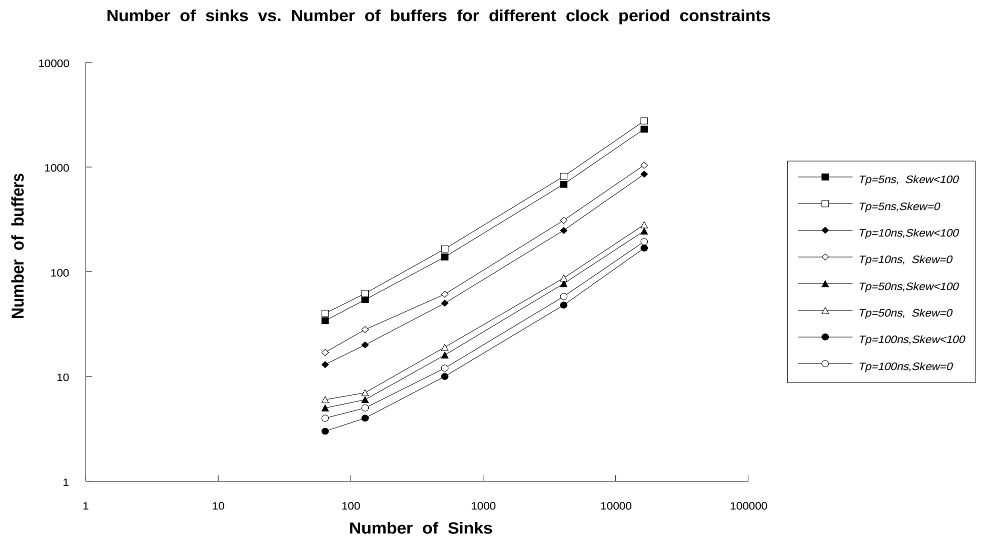

Figure 3: Number of clock pins versus $\mathcal{B}$ for various values of T_P and $\mathcal{S}$.

We obtain constraint versus buffer area, and phase delay curves by sweeping the T_P constraint (or equivalently the C_∞ constraint), and using different buffers. Since these curves are approximately convex, it is then possible to optimize a particular objective with a small sampling of the solution space or by smoothing the trade-off curves and applying optimization techniques to the result.

6 Results and Conclusions

We implemented the **buBufferInsert-BS** algorithm in C++. The program was tested on a SPARC-station 10. The results of the program were tested and verified on randomly generated test cases. SPICE3e2 was used for simulations, experiments and timing data. The MCNC MOSIS 2.0μ technology parameters and models were used to design the buffers and for the simulations. Figure 3 shows results obtained from running the algorithm on trees of a range of sizes (64-16386 pins), with a range of clock period (5-100 *ns*) and skew constraints (100 and 0 skew). The quality of the algorithm is evidenced by the ratio of buffers to tree edges, which does not exceed 0.6. Furthermore, the results shown in the previous sections show that this algorithm is practical, since it can be used to produce solutions with small phase delays, predictable rise times and small skews.

Note: For brevity, we have omitted proofs and other details. Please contact the authors to receive a copy of the full manuscript.

References

[Bak90] H. B. Bakoglu. *"Circuits, Interconnections, and Packaging for VLSI"*, pages 81–112. Addison-Wesley Publishing Co., 1990.

[CC93] N.-C. Chou and C.-K. Cheng. "Wire Length and Delay Minimization in General Clock Net Routing ". In *International Conference on Computer-Aided Design*, pages 552–555. IEEE, November 1993.

[Fri93] E. G. Friedman. "Clock Distribution Design in VLSI Circuits - an Overview". In *International Symposium on Circuits and Systems*, pages 1475–1478, May 1993.

[PMOP93] S. Pullela, N. Menezes, J. Omar, and L. T. Pillage. "Skew and Delay Optimization for Reliable Buffered Clock Trees". In *International Conference on Computer-Aided Design*, pages 556–562. ACM/IEEE, November 1993.

[SK92] Y. H. Shih and S. M. Kang. "Analytic Transient Solution of General MOS Circuit Primitives". *IEEE Transactions on Computer Aided Design*, 11(6):719–731, 1992.

[SW92] N. A. Sherwani and B. Wu. "Effective Buffer Insertion of Clock Tree for High Speed VLSI Circuits". *Microelectronics Journal*, 23:291–300, July 1992.