

Simultaneous Buffer and Wire Sizing for Performance and Power Optimization*

Jason Cong, Cheng-Kok Koh
 {cong, kohck}@cs.ucla.edu
 Computer Science Dept., UCLA

Kwok-Shing Leung
 ksleung@ichips.intel.com
 Intel Corporation

Abstract

In this paper, we study the *simultaneous buffer and wire sizing (SBWS) problem* for delay and power dissipation minimization. We prove the BS/WS relation for optimal SBWS solutions. This relation leads to a polynomial time algorithm for computing the lower and upper bounds of the optimal SBWS solutions, which enables an efficient optimal algorithm for computing optimal SBWS solutions. We have applied the SBWS algorithms to the clock nets in a spread spectrum IF transceiver chip and HSPICE simulations show that our algorithms can reduce skew and power by a factor of 3.5X and 1.6X, respectively, when compared to the manual layout of the clock nets in the original chip.

1 Introduction

As the VLSI fabrication technology advances to submicron device dimension and gigahertz clock frequency, it is important to consider and optimize both device (i.e. transistor/cell) design and interconnect design simultaneously in order to achieve the objective of delay and power minimization. The objective of this paper is to study the problem of buffers and wire sizing for delay and power optimization. This technique is particularly applicable to buffered clock tree optimization since clock tree generally spans the entire chip and has significant interconnect delay, and the clock signals normally operate at the highest frequency of all signals and dissipate a significant amount of power.

Recent studies show that interconnect delay can be reduced by interconnect topology optimization. For example, interconnect topologies such as maximum performance trees [4], A-trees [8], and low-delay trees [1] have been proposed to minimize interconnect delay. Interconnect delay can be further reduced by sizing device and wire. The wiresizing algorithms in [8, 9, 5, 13, 11] can minimize interconnect delay by optimally assigning different wire width to each wire segment in the interconnect design. Recently, [6, 12] explore the possibility of simultaneous driver/gate and wire sizing for performance and power optimization. The works by [14, 11] consider buffer insertion for either performance optimization or power minimization.

In this paper, we study the problem of *simultaneous buffer and wire sizing (SBWS)* for performance and power optimization. In the SBWS problem, we assume that the buffer locations are given and the variables are the buffer sizes and the wire widths. The major contribution of this paper is that we extend the results on simultaneous driver and wire sizing in [6] where no buffer is considered. Analogous to the DS/WS relation in [6], we prove the BS/WS relation between buffer sizing and optimal wire sizing. The relation leads to a polynomial time algorithm for computing the lower and upper bounds of the optimal SBWS solutions, which enables an efficient optimal algorithm for computing optimal SBWS solutions. We have applied the SBWS algorithms to the clock nets in a spread spectrum IF transceiver chip designed at UCLA for wireless multimedia information systems [3]. HSPICE simulations show that our algorithms can reduce power by a factor of 1.6X when compared to the manual layout of the clock nets in the original chip. The optimized clock nets also have smaller maximum clock delay, smaller maximum skew and sharper signal waveform.

2 Problem Formulation

Let T be a buffered routing tree implementing a signal net \mathcal{N} which consists of a set of m sinks $\{N_1, N_2, \dots, N_m\}$, and a set of b buffers $\{D_2, D_3, \dots, D_b\}$ at fixed locations in T , which is formed by a set of wire segments $\{E_1, E_2, \dots, E_n\}$. The signal net is driven by a driver D_1 of a given size d_1 at the source as shown in Figure 1. A sink N_i has a loading capacitance of $c_{N_i}^s$. We assume that each wire segment has a set of discrete choices of wire widths $\{W_1, W_2, \dots, W_r\}$. We use w_{E_i} to denote the width of the wire segment E_i , and d_i to denote the size of buffer D_i . We define the problem of simultaneous buffer and wire sizing (SBWS) for performance optimization as follows:

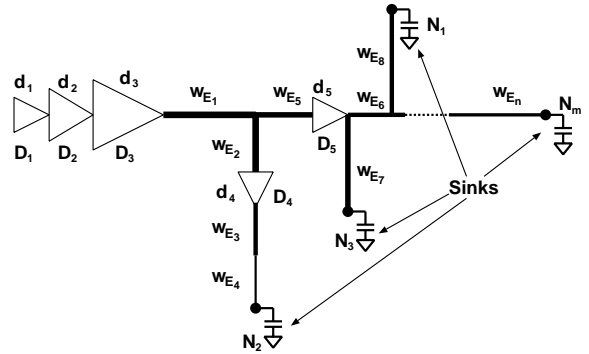


Figure 1: A buffered interconnect tree T with sinks $\{N_1, N_2, \dots, N_m\}$. w_{E_i} denotes the width of the wire segment E_i , $i = 1..n$ and d_i denotes the size of buffer/driver D_i .

Definition 1 Given a buffered routing tree T of a signal net \mathcal{N} with prescribed buffer locations, the SBWS problem for delay minimization is to find a buffer sizing solution $\mathcal{D} = \{d_2, d_3, \dots, d_b\}$ and wiresizing solution $\mathcal{W} = \{w_{E_1}, w_{E_2}, \dots, w_{E_n}\}$ such that the performance measure $t_T(\mathcal{D}, \mathcal{W})$ is optimized.

The performance measure $t_T(\mathcal{D}, \mathcal{W})$ evaluates the “signal delay” of the net from the source driver D_1 to one or several critical sinks, and it is expressed as a linear combination of the signal delays from the source to all sinks [9, 6, 1]:

$$t_T(\mathcal{D}, \mathcal{W}) = \sum_{1 \leq i \leq m} \lambda_{N_i} \cdot t_T^{N_i}(\mathcal{D}, \mathcal{W}) \quad (1)$$

where λ_{N_i} measures the criticality of sink N_i and $t_T^{N_i}(\mathcal{D}, \mathcal{W})$ is the signal delay from the source driver D_1 to sink N_i in the buffered tree T . Note that we use t_T without superscript to denote the weighted sum of delay and $t_T^{N_i}$ with superscript N_i to denote the Elmore delay [10] from source driver to sink N_i .

The weighted-sum formulation can be used in two scenarios. For performance optimization, large λ 's are used for timing critical sinks. For clock skew minimization, clock pins with longer (shorter) delay should be assigned with higher (lower) criticality. In both cases, we normalize λ_{N_i} 's such that $\sum_{i=1..n} \lambda_{N_i} = 1$. The weighted-sum formulation leads to efficient optimal wiresizing algorithm due to the separability, the monotone property, and most importantly, the dominance

*This work is partially supported by ARPA/CSTO under Contract J-FBI-93-112, the National Science Foundation Young Investigator Award, and a grant from Intel Corporation.

property [9]. However, the separability property does not hold for minimizing maximum delay [13] and it is not clear if the dominance property still holds for maximum delay minimization. Although there is a polynomial time algorithm for minimizing maximum delay [11], the actual run-time is long and the memory requirement is prohibitively large (see experimental results in Section 5). [2] shows that by assigning appropriate weight of each sink based on Lagrangian relaxation, the weighted-sum formulation can be used iteratively to minimize maximum delay. Therefore, coupled with Lagrangian relaxation, the SBWS algorithms give an efficient algorithm to minimize maximum delay.

Although buffer and wire sizing are effective approaches to reduce interconnect delay, the drawback is that larger buffer size and additional routing area also increase the power dissipation. In practice, circuit design requires a careful tradeoff between performance and power. We define the SBWS problem for both delay and power optimization (SBWS-DP) as follows:

Definition 2 Given a buffered routing tree T for net \mathcal{N} with prescribed buffer locations, the SBWS problem for both delay and power optimization (SBWS-DP) is to determine a buffer sizing solution \mathcal{D} , and a wire-sizing solution \mathcal{W} , such that the performance measure $obj_T(\mathcal{D}, \mathcal{W})$ defined below is minimized:

$$obj_T(\mathcal{D}, \mathcal{W}) = \alpha \cdot Power_T(\mathcal{D}, \mathcal{W}) + \gamma \cdot t_T(\mathcal{D}, \mathcal{W}) \quad (2)$$

where $t_T(\mathcal{D}, \mathcal{W})$ is the performance measure and $Power_T(\mathcal{D}, \mathcal{W})$ is the power dissipation.

We measure the performance and power dissipation as follows. Note that the formulations and notation used in this paper follow [6] closely.

2.1 Performance Measure for Buffered Tree

For simplicity, we model a buffer as a switch-level RC device, with C_d , C_g and R_{min} being the diffusion capacitance, gate capacitance and resistance of the smallest device, respectively, as in [6]. Note however that our results are not restricted to this simple model only. As long as the output resistance is monotonically decreasing and the input gate capacitance is monotonically increasing as the buffer size increases, the algorithms in this paper still apply since the BS/WS relation only assumes that as buffer/driver size increases, its input capacitance increases and resistance decreases. As in the many works on wiresizing [9, 6, 5], we use the distributed Elmore delay model [10] for interconnect delay measure. The performance measure $t_T(\mathcal{D}, \mathcal{W})$ of an unbuffered routing tree T can be found in [6].

Let T now be the buffered routing tree for net \mathcal{N} . A buffer in T decomposes the net into two subnets. In general, $b - 1$ buffers in T decompose the net into a hierarchy of b subnets. By a top-down topological sorting of the subnets, we can order the subnets as $\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_b$ and define a parent-child relationship among the subnets. Each buffer D_i is a sink with capacitance $C_g \cdot d_i$ of the parent subnet and the driver with resistance R_{min}/d_i and output capacitance $C_d \cdot d_i$ of the child subnet. We order the buffers such that buffer D_i drives subnet \mathcal{N}_i . Also, we denote the unbuffered routing subtree for subnet \mathcal{N}_i by T_i . Note that a chain of b cascaded drivers in the SDWS problem is formulated as b subnets under this new formulation.

Let $Children(D_i)$ denote the set of sinks/buffers driven by D_i directly and we define $\lambda_{D_i} = \sum_{N_j \in Children(D_i)} \lambda_{N_j}$ recursively. Therefore, we can write the performance measure of the buffered tree T as follows:

$$t_T(\mathcal{D}, \mathcal{W}) = \sum_{\text{all } D_i} \lambda_{D_i} \cdot t_{T_i}(\mathcal{D}, \mathcal{W}) \quad (3)$$

where t_{T_i} is the performance measure of the unbuffered tree T_i driven by buffer D_i . The details of the derivation are given in [7].

We can conclude from the above equation that if we are given the buffer sizes, we can obtain an optimal wiresizing solution by applying the technique in [9].

2.2 Trade-Off Between Performance and Power

For simplicity, we consider only capacitive power dissipation in our formulation. However, the results still hold when short-circuit power dissipation is also considered. Given a total capacitance CL driven by the driver, the capacitive dissipation of the single driver of size d is $Power_{cap} = f \cdot CL \cdot V_{dd}^2$, where f is the switching frequency of the input signal [15]. To simplify the expression, we let $\mathcal{L} = f \cdot V_{dd}^2$ and write $Power_{cap} = \mathcal{L} \cdot CL$. Note that CL has three components: the diffusion capacitance of the driver, $C_d \cdot d$, the wire capacitance of the routing tree and the total capacitance of the sinks in the tree. Consider an unbuffered tree T driven by driver D with size d . From [6], the trade-off between performance and power in Eqn. (2) can be written as follows:

$$obj_T(\mathcal{D}, \mathcal{W}) = \alpha \cdot \mathcal{L} \cdot C_d \cdot d + \hat{t}_T(\mathcal{D}, \mathcal{W}) \quad (4)$$

where $\hat{t}_T(\mathcal{D}, \mathcal{W})$ is an expression similar to $t_T(\mathcal{D}, \mathcal{W})$ except for differences in the coefficients. Therefore, when we consider a buffered tree T , the trade-off objective $obj_T(\mathcal{D}, \mathcal{W})$ is

$$\begin{aligned} & \sum_{\text{all } D_i} \lambda_{D_i} \hat{t}_{T_i}(\mathcal{D}, \mathcal{W}) + \alpha \cdot \mathcal{L} \cdot C_d \cdot \sum_{\text{all } D_i} d_i \\ &= \sum_{\text{all } D_i} \left\{ \lambda_{D_i} \hat{t}_{T_i}(\mathcal{D}, \mathcal{W}) + \sum_{D_j \in Children(D_i)} \alpha \cdot \mathcal{L} \cdot C_d \cdot d_j \right\} + \\ & \alpha \cdot \mathcal{L} \cdot C_d \cdot d_1 \end{aligned} \quad (5)$$

Note that the term $\lambda_{D_i} \hat{t}_{T_i}(\mathcal{D}, \mathcal{W}) + \sum_{D_j \in Children(D_i)} \alpha \cdot \mathcal{L} \cdot C_d \cdot d_j$ is again very similar to $\lambda_{D_i} t_{T_i}(\mathcal{D}, \mathcal{W})$ except for differences in the coefficients. Therefore, in the following, we will present the results which assumes optimization of $t_T(\mathcal{D}, \mathcal{W})$ in Eqn. (3) instead of $obj_T(\mathcal{D}, \mathcal{W})$.

3 Properties of Optimal SBWS Solutions

To solve the SBWS problem, we will apply the same approach used to solve the simultaneous driver and wire sizing (SDWS) problem. The optimal SDWS solution is computed by first computing the upper and lower bounds of the optimal solution followed by, if necessary, exploring the extensively pruned solution space using dynamic programming. The key to solving the SDWS problem is the DS/WS relation between a driver sizing solution and a wiresizing solution [6].

3.1 DS/WS Relation of Optimal SDWS Solutions

Given a routing tree T with one or more critical sinks. Let R_d be the resistance of the driver driving the routing tree and \mathcal{W}^* be the corresponding optimal wire width assignment. Consider another driver such that $R'_d (> R_d)$ is the resistance of the driver and \mathcal{W}''^* is the corresponding optimal wire width assignment.

(DS/WS Relation): [6] For any tree T with one or more critical sinks, $R_d < R'_d$ implies \mathcal{W}^* dominates \mathcal{W}''^* . \square

A wire width assignment \mathcal{W} dominates another assignment \mathcal{W}' if for any segment E , the width assignment of E in \mathcal{W} is greater than or equal to that of E in \mathcal{W}' . This result reveals the relation between driver sizing and wiresizing, and it plays an important role in determining the lower and upper bounds of the optimal SDWS solutions. Starting from minimum-width wires, \mathcal{W}_0 , an optimal driver sizing solution, denoted \mathcal{D}_1 is computed. Based on \mathcal{D}_1 , the corresponding optimal wire sizing solution, denoted \mathcal{W}'_1 is then computed. The process of driver sizing followed by wiresizing continues until there is no change in the wire-sizing solution. This process computes the lower bound of the optimal SDWS solution. The upper bound can be obtained by starting with maximum-width wires.

We note that since the drivers are restricted to be at the source of the interconnect tree in the SDWS problem, a change in the driver sizing solution affects the wiresizing solution from the *upstream only* and

a change in the wiresizing solution affects the driver sizes from the *downstream only*. The difficulty of adopting a similar approach to solve the SBWS problem is that a change in the buffer sizing solution affects the buffer and wire sizing solution from the upstream and downstream. Similarly, a change in the wiresizing solution has both upstream and downstream effect on the buffer and wire sizing solution. To use a similar approach to solve the SBWS problem, we present the BS/WS relation between buffer and wire sizing solutions in the following. The proofs are given in [7].

3.2 Properties of Optimal Buffer Sizing Solutions

Consider a buffer D_i with a parent buffer D_j . Assume that we have a wiresizing solution \mathcal{W} and we know the size of D_j and the loading capacitance of the sinks that D_i is driving. We want to compute the optimal size of driver D_i . The variable d_i appears in both $t_{T_i}(\mathcal{D}, \mathcal{W})$ and $t_{T_j}(\mathcal{D}, \mathcal{W})$. Taking differential of $t_T(\mathcal{D}, \mathcal{W})$ with respect to d_i and setting the differential to zero and solving for d_i , we have the following theorem:

Theorem 1 *Given net \mathcal{N}_i and its parent net \mathcal{N}_j . Assuming a wiresizing solution \mathcal{W} and that both the loading capacitances of the sinks in \mathcal{N}_i and the size d_j of the driver of \mathcal{N}_j , namely D_j , are given, the optimal size for the buffer D_i can be computed as follows:*

$$d_i = \left(\frac{\lambda_{D_i} \cdot R_{min} \cdot \left(\sum_{u \in \text{sink}(T_i)} c_u^s + c_0 \cdot \sum_{E \in T_i} w_E + \sum_{E \in T_i} c_1 \right)}{\lambda_{D_j} \cdot C_g \cdot \left(\frac{R_{min}}{d_j} + r_0 \cdot \sum_{E \in P(D_j, D_i)} \left\{ \sum_{u \in \text{sink}(E)} \lambda_u \right\} \cdot \frac{1}{w_E} \right)} \right)^{1/2} \quad (6)$$

where $\text{sink}(E)$ is the set of sinks in T_j which are descendants of edge E , and λ_u is the normalized criticality of sink u in T_j .

This is a generalization of the results for driver sizing of a chain of cascaded drivers in Theorem 1 in [6]. The constant stage ratio in Theorem 1 of [6] can be derived from the above equality since in the case of SDWS, $\lambda_{D_i} = \lambda_{D_j}$, and $r_0 \cdot \sum_{E \in P(D_j, D_i)} \left\{ \sum_{u \in \text{sink}(E)} \lambda_u \right\} \cdot \frac{1}{w_E} = 0$ due to negligible wire resistance between cascaded drivers.

In a similar manner in which the local refinement operation for wiresizing is defined, we define the local refinement operation for buffer sizing as follows: the local optimal size of a buffer is computed using the expression in Eqn. (6) by keeping other buffer sizes and wire sizes intact. Note that the above expression assumed that the buffer size is continuous. However, we can also make the buffer size discrete (e.g., multiples of the minimum-size) as in the case of assuming a finite set of wire widths $\{W_1, \dots, W_r\}$ for wiresizing.

Another observation that we can draw from the local optimal computation of d_i is that if d_j dominates the optimal size of driver D_j , denoted d_j^* , and the wiresizing solution \mathcal{W} dominates the optimal wiresizing solutions, denoted \mathcal{W}^* , then the size of D_i after the local refinement step also dominates the optimal size of D_i , denoted d_i^* . We can generalize the above observation and prove the following:

Theorem 2 (WS/BS Relation) *Consider two wire sizing solutions \mathcal{W} and \mathcal{W}' , if \mathcal{W} dominates \mathcal{W}' , then the optimal buffer sizing solution \mathcal{D} for \mathcal{W} dominates the optimal buffer sizing solution \mathcal{D}' for \mathcal{W}' .*

3.3 BS/WS Relation of Optimal SBWS Solutions

While the relation between the driver resistances and wiresizing solutions is known (DS/WS relation), we want to investigate the relation between the loading capacitances and wiresizing solutions. In the following, we consider a unbuffered routing tree T with one or more critical sinks. Let R_d be the resistance of the driver driving the routing tree and \mathcal{W} be the corresponding optimal wire width assignment. Let c_v^s be the loading capacitance of one of the critical sink of T , i.e. $\lambda_v > 0$. This sink is actually a buffer and it drives a downstream subnet. Consider a similar routing tree except that the loading capacitance of the critical

sink at v is $c_v^{s'}$ ($< c_v^s$) and \mathcal{W}' is the corresponding optimal wire width assignment. We establish the relationship between wiresizing solution and the capacitive load (CL) as follows.

Theorem 3 (WS/CL Relationship) *If $c_v^s > c_v^{s'}$, then there exists a wiresizing solution \mathcal{W} such that for all edges $E \in T$, $w_E \geq w'_E$.* \square

Combining the WS/BS and WS/CL relationships, we can deduce the following relation between buffer sizing and wire sizing solutions:

Theorem 4 (BS/WS Relation) *Consider two buffer sizing solution D and D' , if D dominates D' , then there exists an optimal wiresizing solution \mathcal{W} for D and an optimal wiresizing solution \mathcal{W}' for D' such that \mathcal{W} dominates \mathcal{W}' .*

Analogous to the the DS/WS relation for the SDWS problem, the BS/WS relation plays an important role in determining the lower and upper bounds of the optimal SBWS solutions in the next section.

4 SBWS Algorithms

4.1 Lower and Upper Bound Computation

We first present an algorithm called the SBWS-LU-Bound Algorithm to compute the upper and lower bounds of an optimal SBWS solution for a given buffered interconnect tree: Starting with an initial wire width assignment (say, all wires have the minimum width) and an initial buffer sizing solution (say, all buffers have the minimum size), we perform an arbitrary number of local refinement operations on the sizes of the intermediate buffers based on Theorem 1. Note that we may perform local refinement operations until there is no further changes in the buffer sizes, i.e. the buffer sizes converge for the current set of wiresizing solution (convergence is guaranteed since each local refinement reduces the performance measure), or we have reached a pre-determined upper limit of the number of local refinement operations. Now, we perform a delay optimal wiresizing algorithm [9] on the routing tree of each subnet based on the resistance of the driver of the subnet and the sink capacitances of downstream buffers or loading pins. A new iteration of local refinement operations for buffers are carried out to yield a new buffer sizing solutions. Then, the optimal wiresizing solution will be computed again based on the new buffer sizing solution. Let an iteration be a series of local refinement operations of buffers followed by applying optimal wiresizing algorithm on each subnet. The process is repeated until the wire width do not change in consecutive iterations. The algorithm is described formally in Figure 2.

SBWS LU-Bound Algorithm

```

 $\mathcal{W}_1 \leftarrow \text{Min\_Wire\_Width}; \mathcal{D}_1 \leftarrow \text{Min\_Buffer\_Size};$ 
while true
  Compute new buffer size by Theorem 1:
   $\mathcal{D}_1 \leftarrow \text{iterations of } \text{buffer\_local\_refinement}(\mathcal{D}_1, \mathcal{W}_1);$ 
  Compute new wire size by [9]:  $\mathcal{W} \leftarrow \text{Optimal\_Wiresizing}(\mathcal{D}_1, \mathcal{W}_1);$ 
  if  $\mathcal{W} > \mathcal{W}_1$  then
     $\mathcal{W}_1 \leftarrow \mathcal{W}$ 
  else break;
end while
Output  $(\mathcal{D}_1, \mathcal{W}_1)$  as the lower bound of the optimal SBWS solution;
Upper bound  $\mathcal{D}_u$  and  $\mathcal{W}_u$  can be computed in a similar fashion by
starting with  $\text{Max\_Wire\_Width}$  and  $\text{Max\_Buffer\_Size}$ ;

```

Figure 2: The SBWS-LU-Bound algorithm for computing lower and upper bounds of optimal SBWS solution for a given buffered tree T .

Let \mathcal{W}_0 be the initial minimum width assignment and \mathcal{W}_i denote the optimal wire width assignment obtained in iteration i . Similarly, let \mathcal{D}_0 be the initial buffer sizes and \mathcal{D}_i denote the final buffer sizes at the end of iteration i . It is obvious that in the first iteration, local refinement of buffer size will only cause buffer sizes to revise upward since they are all of minimum size initially. Therefore, \mathcal{D}_1 dominates \mathcal{D}_0 . Similarly, the wire widths in \mathcal{W}_1 , which is computed based on \mathcal{D}_1

dominates \mathcal{W}_0 since \mathcal{W}_0 is the minimum-width assignment. In general, we can show by mathematical induction that $(\mathcal{D}_{i+1}, \mathcal{W}_{i+1})$ dominates $(\mathcal{D}_i, \mathcal{W}_i)$ (according to Theorem 4 on BS/WS relation). Hence, the algorithm will terminate (since there is an upper bound on the maximum wire width). We can also observe that the optimal SBWS solution, denoted $(\mathcal{D}^*, \mathcal{W}^*)$, dominates $(\mathcal{D}_0, \mathcal{W}_0)$. Therefore, applying Theorems 2 and 4 and by induction, it will dominate all $(\mathcal{D}_i, \mathcal{W}_i)$ for all i . In other words, the lower bound is computed correctly. Similarly, we can start with a maximum wire width assignment and maximum driver size assignment and compute the upper bound of the optimal SBWS solution.

The most computationally expensive operation in the algorithm is the wiresizing operations which has the worst case complexity of $O(n^3 \cdot r)$ [9]. However, with the use of bundled refinement operation [5], the wiresizing algorithm runs much faster than $O(n^3 \cdot r)$. Experimental results show that the algorithm terminates after three or four iterations in most cases. In addition, the upper and lower bounds meet for most instances, which implies that the optimal SBWS solution is obtained.

4.2 Optimal SBWS Algorithm

In the case where the bounds computed by the SBWS-LU-Bound Algorithm do not meet, we can compute the optimal solution using the dynamic programming technique. For each buffer, if its bounds do not meet, we obtain a set of discrete buffer sizes within the bounds for the buffer.¹ Let the smallest grid-size possible be λ -unit (not to be confused with the criticality value λ_{D_i} or λ_{N_i}). Then, the buffer sizes for buffer D_i is of the form $\{j \cdot \lambda, (j+1) \cdot \lambda, \dots, k \cdot \lambda\}$ where $j \cdot \lambda \geq \mathcal{D}_l(D_i)$ and $k \cdot \lambda \leq \mathcal{D}_u(D_i)$, where $\mathcal{D}_l(D_i)$ and $\mathcal{D}_u(D_i)$ are the lower and upper bounds for D_i computed by the SBWS-LU-Bound Algorithm.

We present a bottom-up approach to optimally solve the SBWS problem. This algorithm can actually be used to find the optimal SBWS solution without going through the bound computation process if both buffer sizes and wire widths are discrete. However, in this approach, the solution space is huge and may not be computationally feasible. The bound computation allows us to reduce the solution space significantly by narrowing the bounds. This speeds up the Optimal SBWS Algorithms tremendously. In fact, the bounds computed by the SBWS-LU-Bound Algorithm are very tight and they meet in most of the cases.

We use a technique similar to that in [14, 11] to find the optimal SBWS solution. Note that the objective in [14, 11] is to minimize the maximum Elmore delay. Our objective in this paper is to minimize the weighted delay. For an edge e in the buffered tree, we can define $\lambda_e = \sum_{N_i \in \text{sink}(e)} \lambda_{N_i}$ (similar to the definition of λ_{D_i}).² We define $t_e^{N_i}(\mathcal{D}, \mathcal{W})$ to be the Elmore delay from the upstream endpoint of e to sink N_i where N_i is a sink rooted at the downstream endpoint of e . Then, we define $t_e(\mathcal{D}, \mathcal{W}) = \sum_{N_i \in \text{sink}(v)} \lambda_{N_i} \cdot t_e^{N_i}(\mathcal{D}, \mathcal{W})$. Let T_e denote the subtree (within the unbuffered T_i containing e and not the buffered tree T) rooted at the downstream endpoint of e . Let $CL_{T_e}(\mathcal{D}, \mathcal{W})$ be the total capacitance seen at the downstream endpoint of e . For a given $(\mathcal{D}, \mathcal{W})$, we note that $t_e(\mathcal{D}, \mathcal{W})$ can be computed in a bottom-up fashion as follows:

$$t_e(\mathcal{D}, \mathcal{W}) = \lambda_e \cdot r_e \cdot \left(\frac{c_e}{2} + CL_{T_e}(\mathcal{D}, \mathcal{W}) \right) + \sum_{e' \in \text{Child}(e)} t_{e'}(\mathcal{D}, \mathcal{W}) \quad (7)$$

where $\text{Child}(e)$ contains all child edges of e . Note that if e is a driver or buffer, then r_e is the driver resistance and c_e is twice the diffusion capacitance of the driver. If e is a wire segment, then r_e is the wire resistance and c_e is the wire capacitance.

¹As mentioned before, the SBWS-LU-Bound Algorithm can also be applied for the case where the buffer sizes are discrete and not continuous. In this case, the set of discrete sizes for each buffer are those sizes within the bounds computed by the SBWS-LU-Bound Algorithm. For the case of continuous buffer size, we can use the set of discrete buffer sizes based on the restriction imposed by the fabrication technology.

²Note that we distinguish E from e . E is a wire segment in the routing tree and we use e to denote both a wire segment in the routing tree and the RC-segment representing driver or buffer; see the discussion on the edge N_0-N_+ in Section 2 of [6].

(c, t) -Pair-Computation

```

Let  $e'$  be any child edge of  $e$ 
 $S_1 \leftarrow$  set of  $(c, t)$  pairs of  $e$ 
for each remaining child edge  $e'$  of  $e$ 
   $S_2 \leftarrow$  set of  $(c, t)$  pairs of  $e'$ ;  $S \leftarrow \emptyset$ ;
  for  $i \leftarrow 1$  to  $|S_1|$  and  $j \leftarrow 1$  to  $|S_2|$ 
     $c = c_i + c_j$ ;  $t = t_i + t_j$ ;  $S \leftarrow S \cup \{(c, t)\}$ ;
  end for
   $S_1 \leftarrow$  irredundant  $(c, t)$  in  $S$ ;
end for
 $S \leftarrow \emptyset$ ;
for each size  $s$  of  $e$  in  $\{s_e^l, \dots, s_e^u\}$ 
  Calculate  $r_e$  and  $c_e$  of  $e$  accordingly;
  for  $i \leftarrow 1$  to  $|S_1|$ 
     $t \leftarrow t_i + \lambda_e \cdot r_e \cdot (c_e/2 + c_i)$ 
    if  $e$  is a wire segment then
       $c \leftarrow c + c_e$ ;
    else /*  $e$  is a buffer/driver */
       $c \leftarrow C_g \cdot s$  /* input gate capacitance */
    end if
     $S \leftarrow S \cup \{(c, t)\}$ ;
  end for
end for
return irredundant  $(c, t)$  pairs in  $S$ ;

```

Figure 3: An algorithm to compute the set of irredundant (c, t) pairs of e .

For each edge e and a buffer and wiresizing solution, we associate with e a (c, t) pair where c is the total capacitance at the upstream endpoint of e (e.g., $c = CL_{T_e}(\mathcal{D}, \mathcal{W}) + c_e$ if e is a wire segment) and t is the performance measure given in Eqn. (7). Consider two buffer and wire sizing solutions, then we have (c, t) and (c', t') at e . We observe that if $c \leq c'$ and $t' > t$, then (c', t') is sub-optimal. In other words, the corresponding buffer and wire sizing solution of (c', t') is *redundant* and can be ignored. Therefore, the *Optimal SBWS Algorithm* computes for each edge e in the buffered tree T , a set of *irredundant* (c, t) pairs for e (or correspondingly, a set of buffer and wire sizing solutions for the subtree $T_e + \{e\}$). At the end of the bottom-up computation of (c, t) pairs, the root may have a set of solutions. The optimal solution is achieved by choosing the (c, t) pair with the smallest t at the root. Figure 3 shows how the set of (c, t) pairs is computed for an edge e .

In the computation, we use $\{s_e^l, \dots, s_e^u\}$ to denote the set of sizes for e , where s_e^l (s_e^u) is the smallest (largest) size of e . Suppose each wire segment has at most W_R possible choices of wire widths (though the set of wire widths might be different) and each buffer has at most D_R possible buffer sizes (again, the set of buffer sizes might be different) after bound computation using the SBWS-LU-Bound Algorithm. Then, $\{s_e^l, \dots, s_e^u\}$ for edge e is actually of the form $\{j \cdot \lambda, (j+1) \cdot \lambda, \dots, (j+W_R-1) \cdot \lambda\}$ or $\{j \cdot \lambda, (j+1) \cdot \lambda, \dots, (j+D_R-1) \cdot \lambda\}$ depending on whether e is a wire or a buffer, respectively.

From the (c, t) -Pair-Computation Algorithm, we observe that for each buffer, there are at most D_R irredundant (c, t) pairs. For any wire edge e , the total capacitance at e (including e) consists of wire capacitances in T_e and capacitances of sinks (which may be buffers) in T_e . Now, we want to bound the number of irredundant (c, t) pairs of a wire edge. Consider a subnet T_i with n wire edges and m sinks. For any wire edge e , total wire capacitance rooted at e (including e) has at most $n \cdot W_R$ distinct values and the total sink capacitance rooted at e has at most $m \cdot D_R$ distinct values. Therefore, the total capacitance rooted at e (including e) has at most $m \cdot n \cdot W_R \cdot D_R$ distinct values. In other words, there are at most $m \cdot n \cdot W_R \cdot D_R$ irredundant (c, t) pairs at e .

The most computationally intensive process in the Optimal SBWS Algorithm is to combine S_1 and S_2 and return the set of irredundant (c, t) pairs in the (c, t) -Pair-Computation procedure. A straight forward method to compute the set of irredundant (c, t) pairs is to sort the pairs according to their c -value and then go through the sorted list once to discard redundant (c, t) pairs. Combining two sets S_1 and S_2 and then

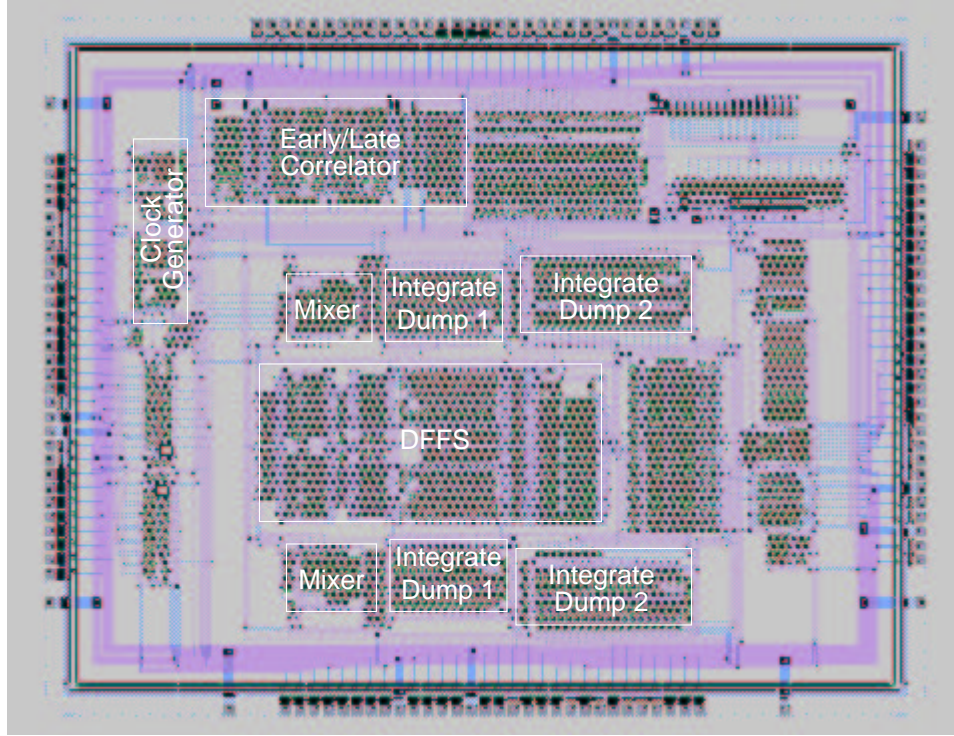


Figure 4: The 12.7 Mchip/s all-digital BPSK direct sequence spread-spectrum IF transceiver chip in $1.2\mu\text{m}$ [3].

sorting the combined set requires $O((m \cdot n \cdot W_R \cdot D_R)^2 \cdot \log(m \cdot n \cdot W_R \cdot D_R))$. For n wire edges, there are $n - 1$ such combinations. Therefore, the Optimal SBWS Algorithm runs in polynomial time of $O(n \cdot (m \cdot n \cdot W_R \cdot D_R)^2 \cdot \log(m \cdot n \cdot W_R \cdot D_R))$.

5 Experimental Results

We apply the SBWS algorithms to the clock nets in a spread spectrum IF transceiver chip developed for wireless adaptive mobile information system (WAMIS) project at UCLA [3]. The integrated single-chip transceiver chip has a die size of $7.9 \times 10.0\text{mm}^2$, is designed under $1.2\mu\text{m}$ 2-level metal SCMOS technology, and has a total power dissipation of 1.1W . The chip is shown in Figure 4. There are two clock nets, named CLK and DCLK, in this design, and they are routed interactively with the Flint place-and-route tool from Berkeley. While the clock skew in each net is important, the designers are more concerned with the inter-clock skew between CLK and DCLK among the registers in the upper and lower Integrate-Dump-1 blocks in Figure 4. Both nets contain registers in the Integrate-Dump-1 blocks.

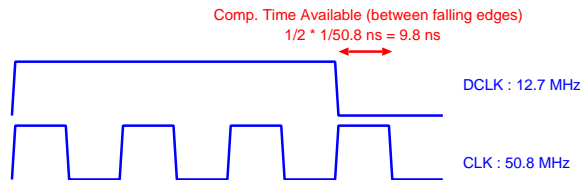


Figure 5: The ideal clock signals of DCLK and CLK.

Both clock signals originate from the Clock-Generator block, with each driven by a chain of 4 cascaded inverters. Each clock signal passes through a buffer block consisting of a chain of 4 cascaded inverters before arriving at registers in the Integrate-Dump-1 blocks. Both the clock source drivers and buffers are designed manually. DCLK operates at 12.7MHz and CLK operates at 50.8MHz . Figure 5 shows the

ideal clock signals of DCLK and CLK, with a computation time of 9.8ns available for the data path in Integrate-Dump-1. However, simulation shows that the worst inter-clock skew of 500ps occurs between the maximum DCLK delay (4.55ns) and the minimum CLK delay (4.05ns) in the upper Integrate-Dump-1 block. The CLK and DCLK nets have a power dissipation of 294.7mW and 25.9mW , respectively.

We apply the SBWS algorithm to speed up the DCLK net. We first assign to each sink (or register) a weight using the following assignment scheme: each sink gets a unnormalized weight which is simply its delay obtained from HSPICE simulation. The weights are then normalized and the SBWS algorithm is applied. Although we have no guarantee that this is a good weight assignment scheme, simulation result shows that it serves the purpose of reducing the inter-clock skew. After optimization, the inter-clock skew is still due to the maximum DCLK delay and the minimum CLK delay of registers in the upper Integrate-Dump-1 block. While the minimum CLK delay remains unchanged, the maximum DCLK delay is reduced to 4.19ns , translating to a factor of 3.5X skew reduction from 500ps to 140ps . A more significant achievement of the SBWS algorithm is that the power dissipation of DCLK is reduced by a factor of 1.6X to 15.5mW after the optimization.

We also apply the SBWS algorithm to optimize the CLK net, assuming that we are only concerned with the skew of the CLK net. The maximum skew of the original CLK is 2018.7ps whereas the maximum skew of the optimized net is 1746.4ps . Besides, the skew within each individual block of the CLK net remains at about the same order as shown in Table 1. Again, significant power reduction is achieved; the power dissipation of the optimized CLK net is reduced by a factor of 2.3X to 127.2mW . The detailed results of the optimized DCLK net is given in Table 2. Also note that all rise and fall times of the clock signals are within 5% of the CLK period. In fact, after SBWS optimization, the clock signals are sharper with smaller rise/fall times.

We also conduct experiments to evaluate the speed-up of run-time due to the lower and upper bound computation in Section 4.1 and the

Block	Original Design			Optimized Design		
	Max-Delay (ns)	Min-Delay (ns)	Skew (ps)	Max-Delay (ns)	Min-Delay (ns)	Skew (ps)
Early/Late-Correlator	3.9057	3.8843	21.4	3.4610	3.4349	26.1
Upper Mixer	4.1514	4.1173	34.1	3.7575	3.7266	30.9
Upper Integrate-Dump-1	4.0523	4.0450	7.3	3.6264	3.6082	18.2
Lower Mixer	4.1650	4.1310	34.0	3.7894	3.7590	30.4
Lower Integrate-Dump-1	4.0657	4.0581	7.6	3.6597	3.6429	16.8
DDFS	5.9030	5.5637	339.3	5.1813	4.9910	190.3
CLK	5.9030	3.8843	2018.7	5.1813	3.4349	1746.4
CLK Power Dissipation (mW)	Original Power: 294.7			Optimized Power: 127.2		

Table 1: Comparison of skews and power dissipation of the manually layout CLK net with the optimized CLK net

Block	Original Design			Optimized Design		
	Max-Delay (ns)	Min-Delay (ns)	Skew (ps)	Max-Delay (ns)	Min-Delay (ns)	Skew (ps)
Upper Integrate-Dump-1	4.5553	4.5489	6.4	4.1897	4.1756	14.1
Upper Integrate-Dump-2	4.5630	4.5502	12.8	4.3088	4.2852	23.6
Lower Integrate-Dump-1	4.4921	4.4856	6.5	4.0979	4.0841	13.8
Lower Integrate-Dump-2	4.6183	4.6050	13.3	4.2257	4.2021	23.6
DCLK	4.6183	4.4856	132.7	4.3088	4.0841	224.7
DCLK Power Dissipation (mW)	Original Power: 25.9			Optimized Power: 15.5		

Table 2: Comparison of skews and power dissipation of the manually layout DCLK net with the optimized DCLK net

quality of the wiresizing solutions obtained by assigning appropriate weights to sinks by iterative Lagrangian relaxation [2] to minimize maximum delay, as compared to the solutions obtained by [11]. Table 3 reports the run-time and the maximum delay by HSPICE simulations of six routed nets in an Intel microprocessor design [5].³ The experimental results show that comparable maximum delay is achieved by our method with significantly shorter run-time. Note that the wires in the six nets are divided into 100 μ m segments before we apply the two algorithms. Finer division of wires leads to prohibitively large memory requirement for the algorithm in [11]. The SBWS algorithm with iterative Lagrangian relaxation does not have such problem.

Net	Optimal Algorithm in [11]		SBWS with Lagrange-Multiplier	
	Delay (ns)	Run-Time (s)	Delay (ns)	Run-Time (s)
net1	0.17145	3.729	0.17145	0.114
net2	0.23557	13.903	0.23562	0.067
net3	0.35121	207.194	0.35121	3.924
net4	0.39744	312.854	0.39374	0.266
net5	0.42730	253.091	0.45300	0.172
net6	0.95751	8966.805	0.95758	19.465

Table 3: Comparison between wiresizing solutions obtained by [11] and the Lagrangian-Based approach for minimizing maximum delay. The wires in the nets are divided into 100 μ m segments before we apply the two algorithms.

6 Conclusion and Future Work

The major contribution of our work is to establish the BS/WS relation between buffer sizing and wire sizing solution, which leads to a polynomial time algorithm to compute the lower and upper bounds of the optimal SBWS solutions and efficient algorithm to compute the optimal SBWS solutions. The results in this paper have shown convincingly that simultaneous buffer and wire sizing can lead to significant reduction in the power dissipation, while achieving better clock delays and clock signals with sharper rise and fall edges.

Acknowledgements

The authors would like to thank Dr. Charles Chien and Prof. Rajeev Jain at UCLA EE Department for providing the transceiver chip, and Mr. Takumi Okamoto and Mr. Lei He for their helpful discussions.

³These are actually multi-source nets. We randomly assign one pin to be the source driver.

REFERENCES

- [1] K. D. Boese, A. B. Kahng, and G. Robins, "High-Performance Routing Trees With Identified Critical Sinks", *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 182-187.
- [2] C.-P. Chen, Y.-W. Chang and D. F. Wong, "Fast Performance-Driven Optimization for Buffered Clock Trees Based on Lagrangian Relaxation," *Proc. ACM/IEEE Design Automation Conf.*, 1996, pp. 405-408.
- [3] C. Chien, P. Yang, E. Cohen, R. Jain, and H. Samueli, "A 12.7Mchip/s All-Digital BPSK Direct Sequence Spread-Spectrum IF Transceiver in 1.2 μ m CMOS," *Proc. IEEE Int'l Solid-State Circuits Conf.*, 1994, pp. 30-31.
- [4] J. P. Cohoon and L. J. Randall, "Critical Net Routing", *Proc. IEEE Int'l. Conf. on Computer Design*, 1991, pp. 174-177.
- [5] J. Cong and L. He, "Optimal Wiresizing for Interconnects with Multiple Sources," *Int'l Conf. on Computer-Aided Design*, Nov. 1995, pp. 568-574.
- [6] J. Cong and C.-K. Koh, "Simultaneous Driver and Wire Sizing for Performance and Power Optimization," *IEEE Trans. on VLSI Systems*, Dec. 1994, pp. 408-425.
- [7] J. Cong, C.-K. Koh and K.-S. Leung, "Simultaneous Buffer and Wire Sizing for Performance and Power Optimization," *UCLA Computer Science Department Technical Report 960017*, Jun. 1996, available at <http://ballade.cs.ucla.edu/~kohck/papers/sbws/sbwstr.ps.Z>.
- [8] J. Cong, K.-S. Leung, and D. Zhou, "Performance-Driven Interconnect Design Based on Distributed RC Delay Model", *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 606-611.
- [9] J. Cong and K.-S. Leung, "Optimal Wiresizing Under Elmore Delay Model," *IEEE Trans. on Computer-Aided Design*, Mar. 1995, pp. 321-336.
- [10] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifier", *J. Applied Physics*, 19(1948), pp. 55-63.
- [11] J. Lillis, C.-K. Cheng, and T.-T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," *Proc. Int'l Conf. on Computer-Aided Design*, Nov. 1995, pp. 138-143.
- [12] N. Menezes, S. Pullela, and L. T. Pillage, "Simultaneous Gate and Interconnect Sizing for Circuit-Level Delay Optimization," in *Proc. ACM/IEEE Design Automation Conf.*, Jun. 1995, pp. 690-695.
- [13] S. S. Sapatnekar, "RC Interconnect Optimization under the Elmore Delay Model," in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 387-391.
- [14] L. P. P. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay," in *Proc. Int'l Symp. on Circuits and Systems*, 1990, pp. 865-868.
- [15] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: a Systems Perspective - 2nd ed*, Addison-Wesley, 1993.