Wiresizing with Buffer Placement and Sizing for Power-Delay Tradeoffs [*]

Jatan C. Shah and Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
201 Coover Hall, Iowa State University
Ames, IA 50011, USA

Abstract

With the increasing influence of the resistive effects of interconnects on the performance of VLSI systems, a greater stress is being laid on careful interconnect design. One prominent technique is the approach of sizing wires for long interconnects to achieve the desired speed and power characteristics [1–4]. It has also been suggested that one may appropriately insert repeaters [5] for significant delay reductions. This paper unifies these approaches to optimizing an interconnect by placing a prespecified number of buffers (drivers *and* repeaters) using a dynamic programming procedure and then performing simultaneous wire and buffer sizing using a sensitivity-based heuristic. Experimental results are presented to prove the utility and performance of the approach.

1 Introduction

With an increasing need for high performance VLSI circuits, a large amount of effort is being applied in optimizing the performance of the digital circuits by means of optimization at the physical level of VLSI design. Specifically, with decreasing minimum feature sizes, long interconnects may have significant resistances, in which case the lumped capacitance model of an interconnect is no longer valid, and the distributed nature of the wire resistance must be taken into consideration.

While an increase in the wire width decreases its resistance, it also increases the capacitance and the dissipated power. Therefore, optimization needs to be carried out by suitably sizing the interconnects to achieve the best power-delay tradeoff.

In [1–3], wire sizing alone was used to reduce interconnect delays. Subsequently, it was shown in [4] that a significant improvement in performance can be achieved by optimizing the sizes of both the wires and the driver at the root of the interconnect tree.

While wire and buffer sizing help reduce delay significantly, the load capacitance for long nets driven by the cascade of drivers at the root is extremely large, thereby imposing impractical buffer width requirements at the root. A solution to this problem has been proposed in [5] wherein buffers in the form of repeaters are introduced (without wire sizing) at different intervals in the interconnect. Introduction of repeaters essentially decomposes the large interconnect into two or more sub-nets which in themselves constitute smaller load capacitances and hence require smaller buffers to drive them.

We are also aware of a parallel effort in this area by Lillis, Cheng and Lin [6]. The approach uses a dynamic programming formulation and reduces the search space using judicious approximations.

In this paper, we solve the problem of inserting drivers and repeaters into an interconnect and sizing them and the wires to derive power-delay tradeoff curves. The tradeoff approach to optimization presented here is supported by the sharply rising nature of the power-delay curves for an interconnect which indicate that the power dissipation corresponding to the minimum delay may not be as desirable a solution as meeting a slightly larger delay specification with a significant reduction in power dissipation.

We present a two step method to optimize the performance of an interconnect under delay constraints at the leaf nodes. The first step involves solving a buffer insertion problem using a dynamic programming formulation for the appropriate introduction of repeaters in the RC wire tree. The second step sizes the inserted buffers as well as the wire segments to meet all of the delay constraints, using a sensitivity-based heuristic.

2 Problem Formulation

2.1 Interconnect and Buffer Modeling

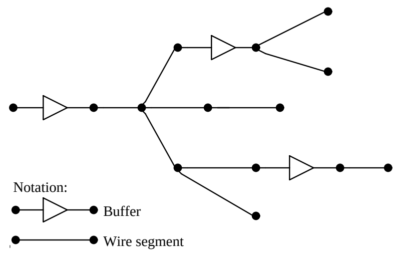

Figure 1: An Example of a Buffered Interconnect Tree

This work assumes that the interconnect net is in the form of a tree, T, as shown in Figure 1, where each branch of the tree is either a buffer or a wire section. Each wire section can further be modeled as a succession of RC wire segments as shown in Figure 2. The resistance, R_i and the capacitance C_i of i^{th} segment of the tree are, given by formulæ$\rho l_i / w_i$ and $\beta l_i w_i$,

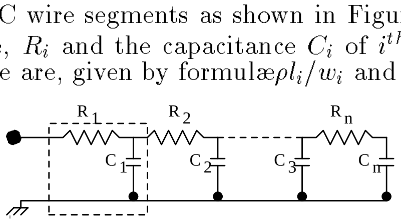

Figure 2: Cascaded RC model of wire segment

where w_i and l_i are, respectively, the width and the length of the i^{th} segment, ρ is the sheet resistance of

[*]This work was supported in part by the National Science Foundation's Faculty Early Career Development Award under contract MIP-9502556.

the wire, and β is the wire capacitance per unit area.

The buffers in the RC tree T are modeled as shown in Figure 3. The input end of the buffer is modeled as a gate capacitance, C_{gi}. The output end of the buffer is modeled by a driver resistance, R_{di}, and the source-drain capacitance. The driver resistance and the gate capacitance are, respectively, given by $\rho_d l_{di}/w_{di}$ and $\beta_d l_{di} w_{di}$, where w_{di} and l_{di} are respectively width and the length of the i^{th} buffer, ρ_d is the resistive constant for the buffer, and β_d is the capacitance per unit area of the channel diffusion. The expression for the source-drain capacitance at the output is similar to that for C_{gi}, with a different proportionality constant.

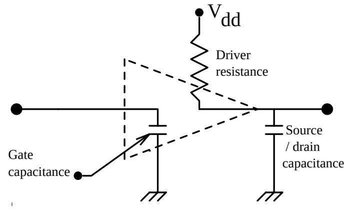

Figure 3: Buffer Model

2.1.1 Delay Modeling

The delay of the interconnect tree is determined by using the Elmore delay model [8]. For an RC tree, the Elmore delay from the root to any node is given by:

$$\sum_{j \in P_i} R_j \Big(\sum_{k \in desc(j)} C_k \Big)$$

where P_i is the path from the root node to that node, and $desc(j)$ is the set of nodes that are descendants of node j in the tree.

The introduction of buffers at intermediate points in an RC tree modifies the delay computation to the leaf nodes since the presence of a buffer isolates the downstream capacitance of that buffer from the rest of the tree. Consequently, the delay equations for the leaf nodes will not include the capacitances that have been isolated by the buffers introduced in the tree, and the introduction of buffers divides the tree T into m subtrees T_i, $1 \leq i \leq m$, each of which is an RC tree, such that the delay to leafnode i is now given by:

$$d_i = \sum_{\forall j \in P_i} R_j \Big(\sum_{\forall k \in desc(j),\, k \in T_i} C_k \Big) \qquad (1)$$

where T_i is the subtree to which the node j belongs.

2.1.2 Power Modeling

The power dissipated in the interconnect tree comprises two components [9]:

(a) The **dynamic** power dissipated by a buffer is the power required to charge/discharge the total capacitance driven by that buffer.
(b) The **short circuit** dissipation of a buffer is the dissipation of power due to the existence of a direct path from V_{dd} to ground during the transient period which is proportional to the rise time of the signal driving the given buffer.

The dynamic power component P_{di}^{dyn}, is given by:

$$P_{di}^{dyn} = \sum_{i=1}^{m} \sum_{j \in T_i} C_j V_{dd}^2 f$$

where C_j is the capacitance at node j, V_{dd} is the supply voltage, f is the switching frequency of the net and T_i is as defined in Section 2.1.1. It can be easily verified that P_{di}^{dyn} may equivalently be written as:

$$P_{di}^{dyn} = \sum_{i=1}^{m} C_{di} V_{dd}^2 f \qquad (2)$$

where C_{di} is the downstream capacitance being driven by the buffer at the root of the subtree T_i.

The short circuit dissipation is modeled as:

$$P_{di}^{sckt} = \frac{\beta}{12} (V_{dd} - 2|V_T|)^3 \tau\, f \qquad (3)$$

where P_{di}^{sckt} is the short circuit power, β is given by $K(w_{di}/l_{di})$, where K is the transconductance, τ is the rise time of the input signal, and $|V_T|$ is the magnitude of the threshold voltage of the transistors in the buffer. Both β and $|V_T|$ are assumed to be equal for n and p transistors. The model estimates the value of τ using the Elmore delay between the two buffers.

A series of experiments conducted with different number of repeaters and different wire sizes for interconnects indicated that under normal working conditions the short circuit power dissipation comprised an insignificant fraction ($\leq 1\%$) of the total power dissipation. Hence, although the short-circuit power is included in the problem formulation, it is ignored in the implementation of the wire and buffer optimization algorithm, leading to a significant reduction in the execution times of the program.

We observe here that when the width of a wire j belonging to a subtree T_i is changed, the effects of the change are localized to T_i. Similarly, whenever a gate or buffer size is changed, the delay of the interconnect network can be recalculated using incremental techniques. These observations are particularly important in ensuring the efficient implementation of the sensitivity algorithm described in Section 4.

2.2 Problem Statement

The problem comprises two steps, namely:

Step 1: Buffer placement in the unsized interconnect network using dynamic programming to position a prespecified number of buffers at the driving end of one or more wire segments which constitute a set of legal positions for buffer insertion. Note that this step is only used to determine the best buffer locations, and that the optimal buffer sizes obtained in this step are discarded in Step 2.

Step 2: Simultaneous wire and buffer sizing using a sensitivity-based heuristic to minimize the delay to the leaf nodes with the minimal increase in the power. Given that the parameters of the problem are the widths of the wire segments in tree, $w_i, i = 1 \cdots n_1$, and the widths of the drivers, $w_{di}, i = 1 \cdots n_2$, in a tree T whose leaf-nodes are specified by the set $leafnode(T)$, the problem can be formally stated as:

$$\text{minimize} \sum_{i=1}^{m} P_{di} \qquad (4)$$
$$\text{subject to } d_j < D_{spec,j} \; \forall \; j \in leafnode(T)$$

where P_{di} is the power dissipated in the i^{th} subtree; d_j and $D_{spec,j}$ are, respectively the delay and the timing specification at leaf-node j of the interconnect tree T.

3 Properties of Problem Formulation

3.1 The Discrete Sizing Problem

The *discrete sizing problem* is the problem of finding optimal wire and buffer sizes to solve the problem formulated in (4) such that the wire widths and buffer sizes may only take on values which are integral multiples of a specified base size, usually the minimum feature size for the technology.

Definition 1: An assignment f for a tree T is an $(n_1 + n_2)$ tuple of numbers $[\, w_1, ...w_{n_1}, w_{d1}, ..., w_{dn_2}\,]$, where n_1 and n_2 are the number of wires and buffers, respectively, in the tree T, and $w_i, 1 \leq i \leq n_1$ and $w_{di}, 1 \leq i \leq n_2$ are the widths of the wires and the buffers, respectively.

Definition 2: Given a routing tree T, a wire width assignment f is a monotonic assignment if $w_p \geq w_c$ whenever wire S_p is an ancestor of S_c, and both wires lie in the same subtree.

The significance of the monotonicity of the problem is emphasized by Theorem 1, the proof of which is similar to that of Theorem 1 in [2].

Theorem 1 (The monotonicity property) Under the Elmore delay model, any optimal wire width assignment f^* must be monotone.

In other words, an optimal set of wire sizes in any subtree T_i, defined in Sec. 2.1.1 must be a monotonic wire width assignment. We emphasize that the monotonicity property is valid only for wire sections in the same subtree, and not across subtrees.

The results in Theorem 1 implicitly assume that the maximum allowable size for each wire is the same, although in situations such as a congested routing region, one may prefer to limit the minimum wire size in that region, causing the monotonicity property to fail. However, the monotonicity property is not critical to the correctness of the work presented here.

We now state a second property that is used by the sensitivity based algorithm.

Theorem 2 Let i be a leafnode, and let P_i be the path from the root node to i. Then the delay from root to node i cannot be decreased by increasing any wire or buffer size that does not lie on P_i.

It may be noted that the separability property used in [1] was shown in [2] to be invalid for the case where the Elmore delay to each leafnode is considered independently.

3.2 Continuous Wire Sizing Problem

The *continuous wire sizing problem* is the problem of finding optimal wire widths to solve the wire sizing problem, such that the wire widths may take on any real value. The approach in this work solves the continuous sizing problem and maps the solution to discrete sizes. It is worth noting that Theorems 1 and 2 are valid for the continuous wire sizing problem too.

Definition 3 *Posynomials*: A posynomial is a function g of a positive variable $\mathbf{w} \in \mathbf{R}^n$ that has the form

$$g(\mathbf{w}) = \sum_j \gamma_j \prod_{i=1}^{n} w_i^{\alpha_{ij}} \qquad (5)$$

where the exponents $\alpha_{ij} \in \mathbf{R}$ and the coefficients $\gamma_j > 0$. It is worth noting that the sum and product of posynomials yield another posynomial.

A posynomial in w has the useful property that it can be mapped onto a convex function in x through the elementary variable transformation,

$$(w_i) = (e^{x_i}) \qquad (6)$$

We now state some properties of the delay and power dissipation in an interconnect, the proofs of which have not been included due to space limitations.

Property 1: *The delay along any path of a buffered RC tree is a posynomial function of the parameters w_i and w_{di} of the problem stated in (4).*

Property 2: *The total power dissipation (dynamic and short-circuit) of a buffer in the tree T is a posynomial function of the parameters w_i and w_{di}.*

Property 3: *The continuous wire sizing problem stated in Section 4 is unimodal i.e, any local minimum of this problem is a global minimum*

To observe this, note that the transformation

$$(w_i, w_{di}) = (e^{x_i}, e^{x_{di}})$$

transforms any posynomial function of the w_i's and w_{di}'s to a convex function of the x_i's and the x_{di}'s. Hence, under this transformation, the objective functions as well as the constraints are convex, and since the mapping function is one-to-one, the optimization problem under consideration is unimodal.

4 Implementation

4.1 Buffer Placement

The implementation of the buffer insertion algorithm for an unsized net is described in [7]. The original algorithm in [7] has been modified to include a constraint on the maximum number of buffers that can be inserted. This is done by dynamically eliminating any possible configuration of the tree which requires more buffers than specified by the upper bound.

The output of the buffer insertion algorithm comprises of buffered RC tree T, where the inserted buffers, are selected from a buffer library (L_B). Each of the buffers in the library is specified by its internal delay (D_{buf}), channel width (w_{di}), and the length of diffusion channel l_{di} of the buffer. Buffer insertion is performed by the insert_buffers procedure which accepts the given wire tree T, the buffer library L_B, and

N_{max}, the maximum number of buffers that can be inserted in the tree, as input parameters. It must be noted that the buffer library L_B is only used to determine the positions of the buffers and the sizes of the inserted buffers are set back to minimum size (0.5 μm) before the wire and buffer sizing is done. This is done because the buffer placement algorithm may produce large buffers that are optimal for the unsized net, but may prove to be excessively large after the buffer and wire sizing is done.

Given that B is the set of legal positions for buffer insertion in the tree T, and N is the number of leaf nodes, the complexity of the buffer placement algorithm is $O(B^2 + N)$.

4.2 The Sensitivity-based Heuristic

The sensitivity heuristic used in [2] performs wire size optimization with the objective of minimizing the area under delay constraints at the leaf nodes.

Initially, all widths are set to the minimum allowable value, corresponding to a small value for the objective function, with the possibility that some constraints may be violated. In each step, the philosophy of the heuristic is to attempt to achieve the best reduction in the amount of constraint violation, with minimum increase in the objective function. To this end, the most sensitive wire or buffer is identified, and its size is bumped up by a bumping factor, F.

The buffered net problem has two sets of parameters, the wire widths $[w_1, \cdots, w_{n_1}]$ and the driver widths $[w_{d1}, \cdots, w_{dn_2}]$. The sensitivity with respect to either of these parameters $x_i = (w_i \; or \; w_{di})$ is given by:

$$S_i = \frac{Delay(size = F \cdot x_i) - Delay(size = x_i)}{\Delta P} \qquad (7)$$

where ΔP is the change in power due to the change in the width of the branch (buffer or wire), and the function *Delay* returns the delay from the root node to the leaf node with the maximum violation, which is referred to as the *current_leaf_node* in the pseudo-code.

The sensitivity of each wire and each buffer along the path from the root to the *current_leaf_node* is computed using Equation (7), and the size of the single wire or buffer with the minimum negative sensitivity (i.e, the negative sensitivity with the largest magnitude) is bumped up by the bumping factor, F. In the inner loop, the algorithm checks whether the branch being considered for sensitivity is a wire or a buffer. If the branch is a buffer, and the sensitivity of the buffer is the most negative so far, then the boolean variable *maxsensitive_isbuffer* is set to true, else it is set to false. In either case the variable *maxsensitivity_branch* is set to the number of the branch having maximum sensitivity for the *current_leaf_node*. At the end of the inner loop execution, both the *maxsensitive_isbuffer* and *maxsensitivity_branch* are used to determine parameter to be bumped. The bumping of the maxsensitive wire/buffer is done by the functions *change_wire_width/change_buffer_width* respectively.

The process continues until no wire or buffer has a negative sensitivity, which gives the solution to the unconstrained problem of minimizing the largest delay to the leafnode under maximum wire and driver width constraints or until the delay specifications at all leafnodes are met. This is the stopping criterion alluded to in the pseudo-code.

```
BEGIN ALGORITHM POINT()
/* Power Optimal Interconnect */
  $L_B$ = Buffer library;
  $N_{max}$ = Maximum number of buffers to be inserted;
  insert_buffers ($T$, $L_B$, $N_{max}$);
  $B_{max}$ = maximum allowable buffer size;
  $F$ = bumping factor;
  while (stopping criterion not met)
    current_leaf_node = leaf node with the
      largest delay violation;
    maxsensitivity = 0;
    maxsensitivity_branch = -1;
    maxsensitive_isbuffer = false;
    for each branch $i$ (wire or buffer)
    that is an ancestor of current_leaf_node
      if the present branch $i$ is a buffer
      and $w_{di} < F \times B_{max}$
      and $S_i$ < maxsensitivity then /*$x_i = w_{di}$*/
        maxsensitivity = $S_{di}$;maxsensitivity_branch = $i$;
        maxsensitive_isbuffer = true;
      else if present branch $i$ is a wire
      and $w_i < F \times w_{Ans(i)}$
      and $S_i$ < maxsensitivity then /*$x_i = w_i$*/
        maxsensitivity = $S_i$;maxsensitivity_branch = $i$;
    if (maxsensitivity_branch == -1) exit;
    if maxsensitive_isbuffer then
      change_buffer_width($i$);
    else change_wire_width($i$);
  MAP()
END ALGORITHM POINT()
```

4.3 Mapping Algorithm

After finding a solution to the continuous wire sizing problem, the heuristic calculates the discrete solution by using a sensitivity based mapping algorithm to round off wire sizes to the next higher or lower integer. The solution to the continuous optimization problem problem provides a lower bound on the solution to the discrete sizing problem, and we have observed that the mapping algorithm described here leads to an insignificant degradation in the quality of the solution. The mapping algorithm is given below.

```
BEGIN ALGORITHM MAP()
  Mark all wires and buffers as unprocessed;
  Mark all leaf nodes as unprocessed;
  While (all leafnodes not processed)
    current_leaf_node = unprocessed leaf node
      with largest delay;
    for each unprocessed branch (wire or buffer)$i$
        that is an ancestor of current_leaf_node
      if (width($i$) is an integer) continue;
      $w_{i+} = \lceil$ width($i$)$\rceil$;$w_{i-} = \lfloor$ width($i$)$\rfloor$
      if ($delay(w_{i+}) - delay$(width($i$))
        < $delay(w_{i-}) - delay$(width($i$) ))
        width($i$) = $w_{i+}$;
      else
        width($i$) = $w_{i-}$;
END ALGORITHM MAP()
```

5 Experimental Results

Experiments were carried out for the following circuit examples using the technology parameters in [4]. The Maxdelay and Minpower figures correspond to a unsized interconnect driven by a minimum-sized driver, and may be used as a reference to evaluate the results.

Interconnect 1 A single wire of length 2cm divided into 50μm sections (Maxdelay = 69.89ns, Minpower = 120.2 pJ/transition)

Interconnect 2 A 12 sink tree with a maximum path length of 2cm to the leaf nodes and 50μm sections (Maxdelay = 139.33ns, Minpower = 247.6 pJ/transition)

For fixed values of the number of inserted buffers (N_{ins}) and the maximum allowable buffer size, a power-delay tradeoff curve was obtained for each circuit. Several such tradeoff curves for Interconnect 1 and Interconnect 2 are, respectively, shown in Figures 4, 5 and Figures 6, 7. The maximum buffer sizes fixed at 30 and 50μm respectively, and each plot provides a comparison of the tradeoff curves obtained for different number of inserted buffers. Note that these curves are incomplete; they do not show the maximum delay point corresponding to the minimum-sized wire with a minimum-sized driver and no repeaters. Tables 1 and 2 show the minimum delay achievable using the algorithm for various values of the maximum buffer size and the maximum number of buffers inserted, for Interconnect 1 and Interconnect 2, respectively.

It can be seen that when a small number of buffers is used, the delay can generally be brought down by increasing the number of buffers. As evidenced in Figures 6 and 7, beyond a certain point, insertion of buffers will cause the timing behavior to deteriorate; however, the minimum achievable delay is smaller when more buffers are used. Although it is not shown here due to space limitations, it is obvious that one cannot add buffers indefinitely and expect to reduce delays; beyond a certain point, no additional improvement is provided by inserting more buffers.

The wire profiles for Interconnect 1 for the tightest constraint case ($D_{spec} = 0.5ns$) are given in Figure 8 and 9. The discontinuities in the wire widths indicate the presence of a buffer; for example, in Figure 8, a buffer is inserted at a distance of 240 segments from the root. Specifically, we note that the farthest repeater from the driver is placed closer to the leafnode, corresponding to the fact that a large load capacitance has to be driven. Note that Interconnect 1 has only one leaf node, and for Interconnect 2, the wire sizes along a path to only one selected leaf node have been shown. The curves also indicate that by placing an upper bound on the buffer sizes (B_{max}), the amount of wiresizing done, as compared to buffer sizing can be controlled.

It is observed that depending on the number of repeaters inserted and the positions in which they are inserted, the performance of the net can be improved significantly over the unsized tree with no repeaters. Further wire and buffer sizing is used to meet the specific delay constraints. Our experiments have also shown that our approach of inserting drivers and repeaters, and sizing these and the wires provides better results than the approach of combined driver sizing with wire sizing, without using repeaters, as in [4].

Table 1: Intct 1: Power Dissipation (pJ/transition)

B_{max}	N_{ins}	Power (pJ/tr.)	Min. Net Delay (ns)	CPU Time(s)
0.5μm (min-sized)	1	120.3	68.76	7.22
	2	120.4	68.20	7.13
	3	120.5	68.00	7.10
30μm	1	123.1	6.30	31.5
	2	131.3	3.24	22.4
	3	141.7	2.13	17.3
50μm	1	123.1	5.60	48.1
	2	128.1	3.08	39.1
	3	138.2	2.00	33.7

Table 2: Intct 2: Power Dissipation (pJ/transition)

B_{max}	N_{ins}	Power (pJ/tr.)	Min. Net Delay (ns)	CPU Time(s)
0.5μm (min-sized)	5	247.8	136.78	6.4
	8	248.1	133.43	6.9
	15	248.3	29.10	7.0
30μm	5	253.1	13.48	8.9
	8	259.7	4.99	9.5
	15	271.0	3.02	7.3
50μm	5	256.1	2.54	12.2
	8	261.9	1.24	13.2
	15	275.1	0.87	9.0

References

[1] J.Cong, K. Leung, and D. Zhou, "Performance-driven interconnect design based on distributed RC model," in *Proc. DAC*, pp. 606–611, 1993.

[2] S. S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," in *Proc. DAC*, pp. 387–391, 1994.

[3] J. Cong and K. S. Leung, "Optimal wiresizing under the distributed Elmore delay model," in *Proc. ICCAD*, pp. 634–639, 1993.

[4] J. Cong and C. K. Koh, "Simultaneous driver and wire sizing for performance and power optimization," in *Proc ICCAD*, pp. 206–212, 1994.

[5] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI.* Reading, MA: Addison-Wesley, 1990.

[6] J. Lillis, C. Cheng, and T. Y. Lin, "Optimal Wire sizing and Buffer insertion for Low Power and a Generalized Delay Model," to appear in *Proc. ICCAD*, 1995.

[7] L. P. V. Ginneken, "Buffer placement in distributed RC-tree networks for minimal elmore delay," in *Proc. ISCAS*, pp. 865–868, 1990.

[8] J. Rubenstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Trans. on CAD*, vol. CAD-2, pp. 202–211, July 1983.

[9] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 468–473, Aug. 1984.

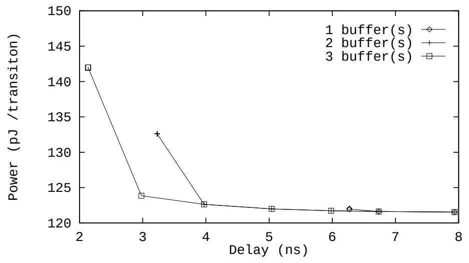

Figure 4: Intct1: Power Delay Curve, $B_{max} = 30\mu m$

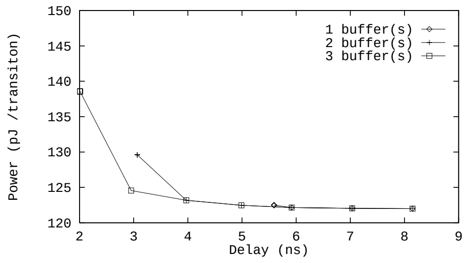

Figure 5: Intct1: Power Delay Curve, $B_{max} = 50\mu m$

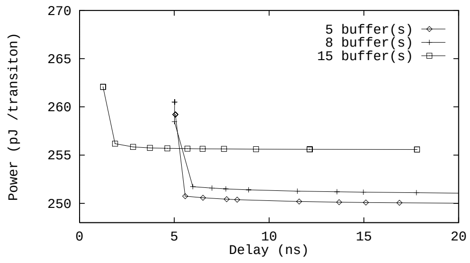

Figure 6: Intct 2: Power Delay Curve, $B_{max} = 30\mu$m

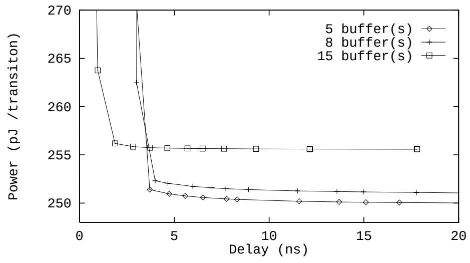

Figure 7: Intct 2: Power Delay Curve, $B_{max} = 50\mu$m

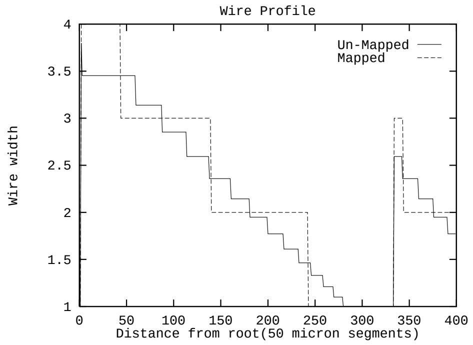

Figure 8: Intct 1: Wire Profile, $B_{max} = 50\mu$m, $N_{max} = 1, Delay = 5.60ns$

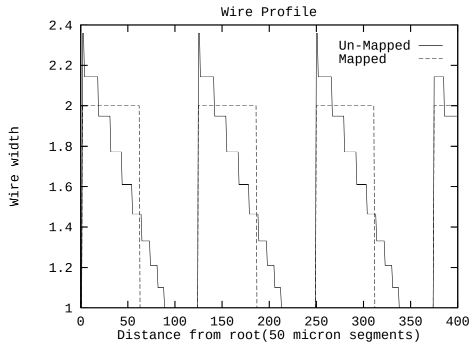

Figure 9: Intct 1: Wire Profile, $B_{max} = 50\mu$m, $N_{max} = 3, Delay = 2.00ns$

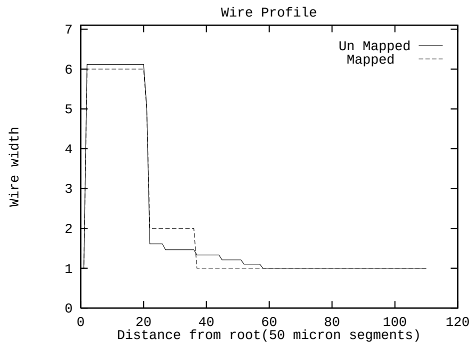

Figure 10: Intct 2: Wire Profile, $B_{max} = 50\mu$m, $N_{max} = 5, Delay = 2.54ns$

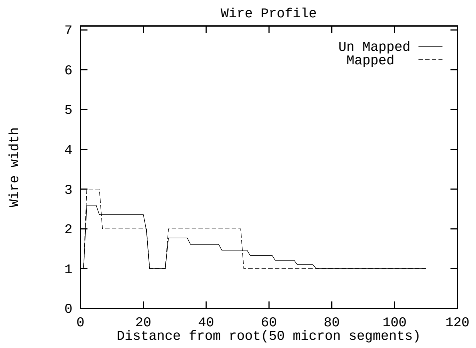

Figure 11: Intct 2: Wire Profile, $B_{max} = 50\mu$m, $N_{max} = 15, Delay = 0.87ns$