Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay

Lukas P.P.P. van Ginneken

International Business Machines Corporation
Thomas J. Watson Research Center
Yorktown Heights, New York

ABSTRACT: This paper presents an algorithm for choosing the buffer positions for a wiring tree such that the "Elmore-delay" is minimal. For given required arrival times at the sinks of the wiring tree, the algorithm chooses buffers such that the required departure time at the source is as late as possible. The topology of the wiring tree, a steiner tree, is assumed given, as well as the possible (legal) positions of the buffers. The algorithm uses a depth first search on the wiring tree to construct a set of time/capacitance pairs that correspond to different choices. The complexity of the algorithm is $O(B^2)$ where B is the number of possible buffer positions. An extension of the basic algorithm allows minimization of the number of buffers as a secondary objective.

INTRODUCTION

Often some gates on a chip have to drive many sinks or long wires. The propagation delay of such a signal can be improved by repowering the signal by little amplifiers, called buffers [1, 2]. The tree structure with the buffers, which distributes the signal over the chip is called the fanout tree. An algorithm to design such fanout trees is described in [3]. However, this algorithm does not take physical information, such as the capacitance and RC effects of the wires, into account.

A large portion of the delay in an integrated circuit is due to the time it takes to charge and discharge the capacitance of the wires and the gates of the transistors. The resistance $R = r\ell$ of a wire increases linearly with its length ℓ and so does its capacitance $C = c\ell$. The RC delay of the wire is $D = 1/2RC = 1/2rc\ell^2$, so it increases quadratically with the length of the wire.

As the scale of integration continues to grow [4] the wire delay will relatively increase and become the dominant factor. Therefore buffering of wiring trees after placement or floor planning will become increasingly important.

The growth of the delay with the length of a wire can be reduced to linear by introducing buffers at fixed distances. In practice, most connections have a tree structure with multiple sinks. In this paper we will present an algorithm for placing the buffers in such a tree structure such that the delay is minimal. The special properties of the Elmore delay model [5, 6] allow the use of a hierarchical algorithm. The algorithm follows the general two phase bottom-up prediction and top-down decision approach [7].

We will show that the complexity of the algorithm is only linear in the number of possible positions of the buffers. A simple extension of the algorithm allows optimization of the cost of the buffers used.

THE BUFFER PLACEMENT PROBLEM

The task of the new algorithm is to put buffers into an existing wiring tree. On the wiring tree there are legal positions where the algorithm may place a buffer. In the actual implementation only global placement and routing information is used, so that small modifications like placing a buffer are allowed.

The capacitance C_i of each sink i and the required arrival time T_i of the signal at each sink are given. Each buffer is characterized by an input capacitance C_{buf}, an internal delay D_{buf}, and an output impedance R_{buf}. A wire with length ℓ has a distributed resistance $r\ell$ and a distributed capacitance $c\ell$.

Let D_i be the delay between the source of the signal (the root of the tree) and the sink i. The algorithm places buffers such that the resulting required departure time T_{source} of the signal at the source is as late as possible. The required departure time can be expressed as:

$$T_{source} = \min_i(T_i - D_i)$$

DELAY CALCULATION FOR WIRING TREES

The wiring trees are modeled by a tree of distributed RC sections [8]. The capacitance to the surrounding wires is modeled as an extra contribution to the capacitance to ground. The root of the tree is the source of the signal, and the leafs of the tree are the sinks of the signal.

Computing the delay of a wiring tree exactly is difficult, and requires the solution of a set of differential equations for the distributed RC sections. It is however possible to derive easily computed estimates for the delay. The pivotal paper [6] presents some easily calculated bounds on the waveform of the step response in a distributed RC tree network; [9] shows that the bounds are also valid for RC meshes and [10] extends the bounds in [6] with non-linear resistors. [6] also presents a simple calculation for RC tree networks of the "Elmore delay" [5]. The Elmore delay has been extended for nonlinear resistors [11] and for an initial charge condition [12]. For reasons of algorithmic complexity, we will use the Elmore delay as the objective function of our algorithm.

The Elmore delay is defined as the first order moment of the impulse response h(t), also known as the inertia:

$$D = \int_0^\infty h(t)tdt$$

CH2868-8/90/0000-0865$1.00 © 1990 IEEE

The model that we consider preserves charge (no leakage to ground). If this model is excited with a negative step function, the Elmore delay is the average arrival time of the electrons at the sink capacitor.

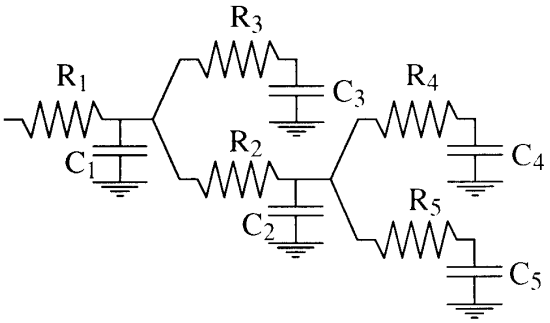

Figure 1. RC tree network with discrete elements

Consider an RC tree network with k resistors and k capacitors (Figure 1). Let C_k be the capacitance of node k and R_k the resistance of branch k. Let $\pi(k)$ be the set of nodes on the path from the root to node k. Let R_{ki} be the resistance of the common sections of the path from k to the root and the path from i to the root: $R_{ki} = \Sigma_{n \in \pi(k) \cap \pi(i)} R_n$ (For instance in Figure 1 $R_{4,5} = R_1 + R_2$). The Elmore delay from the root to sink i is a sum over all sections k in the tree:

$$D_i = \sum_k R_{ki} C_k$$

The delay through a single resistor is the product of its resistance times all the capacitance that is charged through it. Let L_k be the total capacitative load in the sub-tree rooted at k.

$$D_i = \sum_{k \in \pi(i)} R_k L_k$$

For distributed RC lines the summation should be replaced by integration over the length of the line. For uniform lines we can simply replace the distributed capacitance by a lump capacitor in the middle of the line (Figure 2).

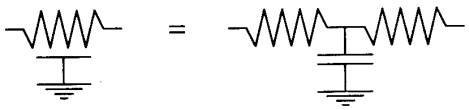

Figure 2. A distributed section and its replacement circuit

Using the Elmore delay model, T_0 can easily be computed recursively. A sub-tree rooted at k can be modeled by two numbers: the required time T_k if the sub-tree would be driven by a buffer of zero impedance, and the load of the sub-tree L_k. When a piece of wire of length ℓ is added at the root of the sub-tree, the new values can be computed by

$$T'_k = T_k - r\ell L_k - 1/2rc\ell^2 \quad (1a)$$

$$L'_k = L_k + c\ell \quad (1b)$$

When a sub-tree is buffered the new values can be computed by

$$T'_k = T_k - D_{buf} - R_{buf} L_k \quad (2a)$$

$$L'_k = C_{buf} \quad (2b)$$

When two sub-trees respectively rooted at n and m are joined by node k, the new values for the whole sub-tree rooted at k are

$$T_k = \min \{T_n, T_m\} \quad (3a)$$

$$L_k = L_n + L_m \quad (3b)$$

THE ALGORITHM

The buffer placement algorithm works by computing the delay using the recursive formulas above. The buffer placement algorithm computes all the different (T_0, L_0) pairs for possible assignment of buffers. We will call these pairs options.

The combinations of the options is done according to equations (3a) and (3b). An option is strictly worse if the load is larger and the required time is earlier. Because some of the options are strictly worse then others, they need not be saved.

When the load is represented as a function of the required time, the process can be viewed as the addition of two functions (Figure 3). Obviously, the number of steps in the new function is not larger than the sum of the number of steps in both terms. We will show later that although the number of possible buffer assignments is 2^B, where B is the number of legal positions for buffers, the number of options at the root cannot exceed B + 1.

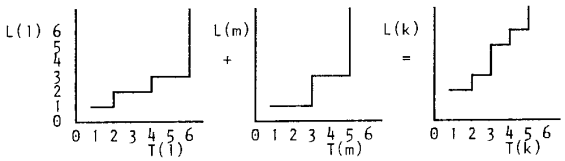

Figure 3. Addition of the loads

The algorithm consists of two phases. During the first phase the function "bottom_up" computes all options for each node in the tree. These options are stored in a global data structure for the second phase. For the options for the root of the tree, the actual delay is calculated, using the output resistance R_{gate} of the gate which produces the signal: $T_{source} = T_0 + L_0 R_{gate}$. The option with the best T_{source} is chosen. The second phase traces back the computations of the first phase that led to this option. While it does that, it places buffers. This phase is similar to the first phase except that now the options are reconstructed to determine which buffer placement led to the optimal solution.

```
1  function bottom_up(k);
2  begin if isleaf(k)
3        then Z: = {(L_k, T_k)}
4        else
5        begin (* compute options for sub-trees *)
6              Z1 := bottom_up(left(k));
7              Z2 := bottom_up(right(k));
8              (* join two sub-trees *)
9              Z := ∅;
10             for p ∈ Z1 ∪ Z2 do
11             begin if ∃T_q ≥ T_p then
12                   begin L: = L_p + min(L_q | T_q ≥ T_p);
13                         Z: = Z ∪ {(L, T_p)};
14                   end;
15             end;
16             (* add wire delay of length ℓ(k) *)
17             for z ∈ Z do
18             begin T_z: = T_z − 1/2rcℓ²(k) − rℓ(k)L_z;
19                   L_z: = L_z + cℓ(k);
20             end;
21             (* add buffer option *)
22             T: = min_{z∈Z}(T − D_buf − R_buf L_z);
23             Z: = Z ∪ {(C_buf, T)};
24             return Z;
25       end;
26 end;
```

The function "bottom_up" first check whether node k is a leaf node (line 2). If not, the function is called recursively to compute the options for the right and left sub-trees (line 6-7). The options are combined by adding the loads of both sub-trees (line 12) for each different required time (line 10-15). Then for each option the RC influence of wire segment k is is calculated (line 17-20). An extra option is finally added for an optional buffer at the root of the sub-tree (line 22-24). In this implementation the legal positions for the buffers are directly after the branching points of the tree. This is done to be able to unload the critical path as much as possible.

The complexity of the algorithm is quadratic in the number of legal positions for the buffers and the number of leafs. Consider the three kinds of operations. Joining two sub-trees gives at most the sum of the options of the sub-trees. Adding a wire to the root of a sub-tree adds no options. Adding an optional buffer to a sub-tree adds one option. Therefore the number of options at the root of the tree will be B + 1. Since there are some loops that iterate through the options in the procedure the complexity is quadratic: $O(B^2 + N)$, where N is the number of leafs in the tree. Notice that the minimization in line 12 does not need a loop if the options are stored in ordered by time.

Many technologies have a limit on the fanout that a gate or a buffer may drive. This limit is specified as a capacitance limit. This limit is easily taken into account by eliminating the options that violate the fanout limit.

In addition to the optimization of the timing, the number of buffers used can be optimized. This is done by using triples of numbers rather than pairs for the options. Each option has in addition to the required time and the load also a number for the cost of the solution. When comparing options, an option can only be discarded if it is worse in all three respects. Unfortunately this makes the algorithm no longer polynomial. However the experiments show that the number of options that results from this extra objective is small.

Another extension of the algorithm allows for different kinds of buffers. Some buffers have a larger internal delay, or a larger input capacitance, but they have a smaller output impedance. A set of different buffers can be used to generate different new options for a buffer position.

RESULTS

The algorithm was implemented as a part of IBM's Logic Synthesis System and it was coded in PL/I. The placement information was obtained from a global placement tool. The placement tool partitions the chip in a rough grid of rectangular areas.

The global routing was done by a steiner tree heuristic, without taking congestion into account. Global routing and placement of the buffers are done on a net by net basis since the database cannot store the wire topology.

The topology of the wires was unknown during the timing analysis of the design, but the placement of the gates and buffers was known. Therefore a lower bound for the Elmore delay was used which does not depend on the topology of the wire. The lower bound consists of two terms.

The first term is the delay resulting from the lump capacitance of the wire. This capacitance is directly proportional to the wire length. The wire length is estimated as half the perimeter of the smallest box containing all pins of the wire. This, multiplied by the output impedance of the gate is the first term of the delay.

To take the resistance into account, the RC-delay of an unbranched line to each sink is added. This is an optimistic (low) estimate of the delay due to the resistance in the wires. Let ℓ_i be the distance from the source to sink i then

$$D_i \simeq R_{gate} L_0 + 1/2rc\ell_i$$

A test example of several thousand CMOS standard cells with wires of up to approximately 1cm long, filled with random logic was used for the experiments. The chip was partitioned in a grid of 6×10 rectangular areas. Using the above method to compute the chip cycle time, the new algorithm produced a result that was 6.8% better than the result of a heuristic that did not use placement information. This number is expected to grow with increase of the scale of integration.

The following table gives a list of pairs of primary inputs and and primary outputs of logic with the worst delay. Delays are expressed as a percentage of the worst chip delay. The table shows that the improvements are consistent, and range from 0.3% to 10.0%. Notice that the worst pair remained the worst after the buffer placement.

WORST DELAY PIN PAIRS				
PRIMARY INPUT	PRIMARY OUTPUT	OLD DELAY	NEW DELAY	GAIN
A0	A0	100.0	93.2	6.8
A0	B0	97.1	90.3	6.8
A0	C0	94.5	89.7	4.8
A0	C4	94.5	89.4	4.8
A0	C5	94.5	89.4	4.8
B0	D0	93.9	89.8	4.1
A0	C1	93.7	88.6	5.1
A0	C2	93.7	88.6	5.1
A0	C3	93.7	88.6	5.1
A0	C6	93.6	88.5	5.1
A0	C7	93.6	88.5	5.1
A0	E0	93.0	88.4	4.6
B0	F0	92.1	88.0	4.1
A0	B4	88.8	83.9	4.9
C0	L0	87.8	75.8	2.0
B0	G5	86.1	82.4	3.7
B0	G6	85.5	81.6	3.9
B0	H1	85.5	81.8	3.7
B0	H2	85.0	81.3	3.7
B0	I0	85.0	81.3	3.7
B0	H3	84.7	81.0	3.7
B0	J0	84.2	83.1	1.1
B0	B5	84.1	83.8	0.3
C0	K1	83.7	73.7	10.0
C0	K0	83.7	75.8	7.9

When also the cost of the solution is included in the optimization, the worst case complexity of the algorithm is no longer polynomial. Figure 4 shows however that the growth of the number of options with the number of sinks is limited. In practice the complexity of the algorithm seems to be limited by $O(n^2)$

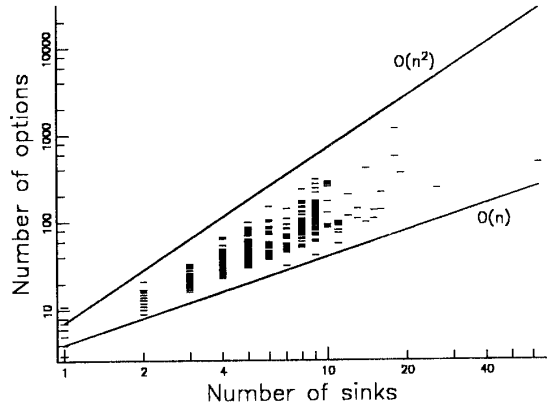

Figure 4. Growth of the number of options

CONCLUSIONS

We presented an algorithm that will place buffers on an existing wiring tree. The algorithm is not a heuristic. Within the Elmore delay model and a finite set of legal buffer positions it finds the optimal solution for a given tree topology. The special properties of the Elmore delay model allows the use of a hierarchical algorithm.

The complexity of the algorithm is polynomial: $O(B^2 + N)$ where B is the number of legal positions for the buffers and N is the number of leafs in the tree.

The algorithm takes the RC effects due to long wires fanout constraints into account. It can handle different kinds of buffers and it can minimize the number of buffers.

A possible future extension of the algorithm is to allow inverters as buffers. This would allow the signal and its inverse to be distributed on a single wire, thus reducing the wiring congestion. This would require an extra bit for each option to indicate the polarity of the signal.

REFERENCES

[1] D. Stroehle, "Avoiding the pittfalls in CMOS design," *New electronics*, vol. 20, no. 12, p. 30, June 1987.

[2] K. Keutzer, "Timing optimization in a logic synthesis system," in G. Saucier, editor, *Proc. int. workshop on logic and arch. synthesis for silicon*, Grenoble, France: Inst. nat. polytechnique, May 1988.

[3] C.L. Berman, J.L. Carter, and K.F. Day, "The fanout problem: from theory to practice," *Proc. Caltech VLSI conf.*, pp. 69-99, 1989.

[4] W.C. Holton and R.K. Cavin, "A perspective on CMOS technology trends," *Proc. of the IEEE*, vol. 74, no. 12, pp. 1646-1668, 1986.

[5] W.C. Elmore, "The transient response of damped linear networks," *Journal of applied physics*, vol. 19, pp. 55-63, Jan 1948.

[6] J. Rubinstein, P. Penfield Jr., and M.A. Horowitz, "Signal delay in RC tree networks," *IEEE trans. on computer aided design*, vol. CAD-2, no. 3, pp. 202-211, July 1983.

[7] L.P.P.P. van Ginneken, *The predictor-adaptor paradigm*, PhD thesis, Eindhoven university of technology, Eindhoven April 1989.

[8] I.T. Ho and S.K. Mullick, "Analysis of transmission lines on integrated circuit chips," *IEEE J. solid-state circuits*, vol. SC-2, no. 4, pp. 201-208, December 1967.

[9] J.L. Wyatt Jr., "Monotone sensitivity of nonlinear nonuniform RC transmission," *IEEE trans. on circuits and systems*, vol. CAS-32, no. 1, pp. 28-33, January 1985.

[10] Q. Yu and O. Wing, "Waveform bounds of nonlinear RC trees," *Proc. int. symp. circuits and systems*, pp. 356-359, Montreal, Canada, May 1984.

[11] P.K. Chan, "An extension of Elmore's delay and its applications for timing analysis of MOS pass transistor networks," *IEEE trans. on circuits and systems*, vol. CAS-33, no. 11, pp. 1149-1152, November 1986.

[12] T.-M. Lin and C.A. Mead, "Signal delay in general RC networks," *IEEE trans. on computer aided design*, vol. CAD-3, no. 4, pp. 331-349, October 1984.