EINDHOVEN UNIVERSITY OF TECHNOLOGY

Computing the entire area/power consumption versus delay tradeoff curve for gate sizing with a piecewise linear simulator

Citation for published version (APA):
Berkelaar, M. R. C. M., Buurman, H. W., & Jess, J. A. G. (1996). Computing the entire area/power consumption versus delay tradeoff curve for gate sizing with a piecewise linear simulator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *15*(11), 1424-1434. https://doi.org/10.1109/43.543774

DOI:
10.1109/43.543774

Document status and date:
Published: 01/01/1996

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 23. Aug. 2022

Computing the Entire Active Area/Power Consumption versus Delay Tradeoff Curve for Gate Sizing with a Piecewise Linear Simulator

Michel R. C. M. Berkelaar, Pim H. W. Buurman, and Jochen A. G. Jess

***Abstract*— The gate sizing problem is the problem of finding load drive capabilities for all gates in a given Boolean network such, that a given delay limit is kept, and the necessary cost in terms of active area usage and/or power consumption is minimal. This paper describes a way to obtain the entire cost versus delay tradeoff curve of a combinational logic circuit in an efficient way. Every point on the resulting curve is the global optimum of the corresponding gate sizing problem. The problem is solved by mapping it onto piecewise linear models in such a way, that a piecewise linear (circuit) simulator can do the job. It is shown that this setup is very efficient, and can produce tradeoff curves for large circuits (thousands of gates) in a few minutes. Benchmark results for the entire set of MCNC '91 two-level examples are given.**

I. Introduction

A. The Gate Sizing Problem

THE PROBLEM TREATED in this paper is the problem of gate sizing. It can be defined as assigning load drive capabilities to the gates in a Boolean network such, that a given delay limit is obeyed, and the total cost in terms of active area and/or power consumption of the circuit is minimal. The problem is very similar to the transistor sizing problem. The main difference is that in gate sizing all transistors in a logic gate are sized simultaneously, whereas transistor sizing sizes single transistors. The main reason we focus on gate sizing is the fact that we want to be able to optimize large circuits. Our ideas are, however, also applicable to transistor sizing.

B. Previous Work

In the past, several algorithms have been proposed to solve the transistor sizing problem. There are a number of heuristic or combined algorithmic/heuristic optimizers [9], [11], [14], [15], [21], and [23] which solve the problem but cannot guarantee optimality of the solution. More recently a number of solutions which use efficient forms of solving a nonlinear programming problem [17], [18], and [22] have been published. They prove that in many cases the heuristic solutions stay far from the global optimum. The most important problems in these nonlinear programming approaches are usually the run time, which becomes too long for large examples, and the convergence. In [1] and [3], a solution to the gate sizing problem was proposed using linear programming (LP), and piecewise linear (PL) approximations of the nonlinear delay formulas. This proved to be fast and, therefore, feasible for large circuits.

Manuscript received January 7, 1994; revised September 21, 1994. This paper was recommended by Associate Editor M. Fujita.

M. R. C. M. Berkelaar and J. A. G. Jess are with the Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands.

P. H. W. Buurman is with Xirion BV, de Meern, The Netherlands.

Publisher Item Identifier S 0278-0070(96)07349-6.

C. Why Compute the Entire Tradeoff Curve?

All of the above approaches can find one point in the solution space of the problem per invocation of the program. Very often, however, a designer is interested in (part of) the tradeoff curve, to be able to compare the advantages and disadvantages of several possible implementations. If we examine typical tradeoff curves as in Figs. 3–5 it is immediately apparent that most designers will want to avoid using solutions in the leftmost parts of the curves, where small gains are made at large costs.

To obtain the tradeoff curves, one could calculate a lot of points by repeatedly using a single-point optimization algorithm. This is not only computationally expensive, but it is also not easy to determine which points are needed to make linear interpolation between them an accurate estimation of the real tradeoff curve. It is just this task that our circuit simulator with variable integration step size performs very efficiently. By introducing a time-dependent delay, (part of) the tradeoff curve is visited during the simulation. Because the different solutions for two slightly different delays are usually close to each other, the new solution can be calculated with only a few updates. We will show that the LP problem of [1] can be mapped onto piecewise linear models, which can be solved with the piecewise linear simulator PLATO. As we shall see in the results section, the PL simulator setup can calculate entire tradeoff curves in about the same amount of CPU time as it takes to solve one LP problem for one solution point.

D. Setup of This Paper

In Section II, we will discuss the LP formulation of the gate sizing problem, summarizing [1] and [3], chs. 4 and 6. In

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

Section III, we will introduce the piecewise linear simulator PLATO. In Section IV, we will discuss the mapping of the gate sizing problem onto piecewise linear models suitable for PLATO. In Section V, we will present results for all MCNC 91 benchmark examples, comparing the [1] approach with the performance of PLATO. In Section VI, we will discuss some numerical aspects of optimizing very large circuits. In Section VII, we will give some conclusions and directions of future work.

II. LP Formulation of Gate Sizing

A. Basic Delay Model of a Gate

To describe the way we have modeled the gate sizing optimization problem for the PL simulator, we will introduce a simple gate level delay model, as introduced in [1]. This is a model which uses the worst case delay of a logic gate, both with respect to inputs as with respect to rise or fall time, as delay value. Although this kind of model can give quite accurate delay estimations for large circuits, it is not very accurate for many small circuits. It is, however, important to realize that any convex delay model can be used with our gate sizing method. In [9] the important class of distributed RC models is proved convex.

We use the following symbols:

- τ delay of a gate
- C capacitance value
- c constant
- S speed factor

We start with the widely used basic model

$$\tau_{gate} = \tau_{int} + c \times C_{load} \tag{1}$$

$$C_{load} = C_{wire} + C_{tr} \tag{2}$$

where C_{wire} is the wire capacitance and C_{tr} is the sum of the capacitances of the connected gates. Because we are dealing with gates of variable size, we introduce the *speed factor* S_{gate} of a gate

$$\tau_{gate} = \tau_{int} + \frac{c \times C_{load}}{S_{gate}}. \tag{3}$$

The fact that it is reasonable to model the load drive capability of a gate by just one parameter S can be justified by the following reasoning: cells are made faster by increasing the gate width of the internal transistors. Their load drive capability grows linearly with this width, as long as we keep the sizing within reasonable limits. The internal capacitances of the cell, however, will also increase almost linearly with the width of the gate. The combination of these two effects will keep the internal delay τ_{int} almost constant over a range of load drive capabilities, but will decrease the delay due to capacitive loading linearly.

Because S_{gate} is implemented in the physical layout by multiplying the width of the transistors, the gate capacitances also grow almost linearly with it, and we get

$$C_{tr} = \sum_{i \in fanout(gate)} S_i \times C_{in,i}. \tag{4}$$

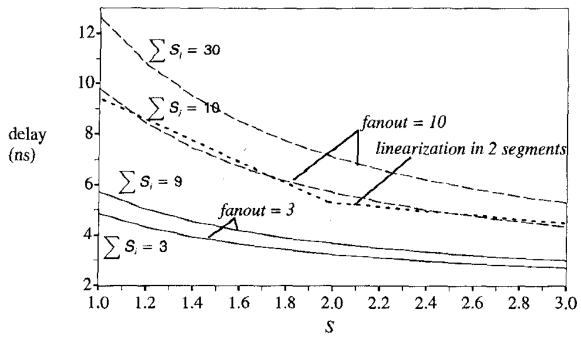


Fig. 1. Delay of a NAND gate under different loads versus its speed factor.

Combining (2)–(4) leads to a complete (nonlinear) delay model

$$\tau_{gate} = \tau_{int} + c \times \frac{C_{wire} + \sum_i S_i \times C_{in,i}}{S_{gate}}. \tag{5}$$

To use this delay model in a LP environment, it has to be linearized. To obtain the desired accuracy, we can use a piecewise linear fit with as many pieces as desired. In [3], it is shown that 3 pieces is enough for above delay model, if we limit S to the range [1, 3]. Increasing the number of pieces beyond that does change the accuracy only marginally ($< 1\%$). Fig. 1 shows a typical case of a 2-input nand gate in a 1.5 m CMOS process under various load conditions. It also shows a possible piecewise linear approximation of one of the curves in 2 pieces.

B. Delay of a Logic Circuit

We model the total delay of a logic circuit by calculating the longest path in the Boolean network [4]. The false path problem is ignored this way. To be able to calculate the longest path, we introduce the symbol T_{gate}, the *schedule time* of a gate. It is defined as the worst case time after which the output of the gate will become stable on an input transition. The arrival times of the primary input signals are assumed to be known. The schedule time of a gate can be expressed in the schedule times of its input signals and its own delay

$$T_{gate} = \tau_{gate} + \max_{i \in inputs(gate)} T_i. \tag{6}$$

Both the summation and the max function can easily be expressed in LP terms.

The wire capacitance C_{wire} in (2) and (5) is estimated based on statistical data from actual layout. See [3] for details.

C. The Total Active Area of a Circuit

Because the transistor sizes in the gates are adapted in just one dimension with changing speed factor, the total active area of a circuit is a linear combination of the speed factors, where the constants c_i reflect the relative contribution to the active area of gate i.

$$A = \sum_{i \in gates} c_i S_i. \tag{7}$$

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

D. CMOS Dynamic Power Consumption

The following discussion assumes that the average switching frequency of the gates does not change during gate sizing. This is, of course, not completely correct: the amount of glitching in the circuit will in general change, and probably become less as the delay differences in the circuit get smaller. Yet, the model presented here to estimate power consumption will be accurate enough if the amount of glitching in the circuit does not change a lot, or is not a large percentage of the total switching activity.

If we define $\overline{f}_{sw}$ to be the average switching frequency of a gate, the dynamic power consumption of a single CMOS gate can be expressed as

$$\begin{aligned} P_{gate} &= \overline{f}_{sw} \times \frac{V_{dd}^2}{2} \times C_{load} \\ &= c_p \times C_{load} \\ &= c_p \times (C_{wire} + C_{tr}) \\ &= c_p \times \left(C_{wire} + \sum_{i \in fanout} S_i \times C_{in,i} \right). \end{aligned}$$

The total power consumption of a circuit is the sum of the power consumptions of the individual gates

$$\begin{aligned} P &= \sum_{i \in gates} P_i \\ &= \sum_{i \in gates} c_{p,i} C_{wire,i} + \sum_{i \in gates} c_{p,i} \sum_{j \in fanout} S_j C_{in,j}. \end{aligned} \quad (8)$$

The first summation of (8) represents the power consumed while charging and discharging the capacitances of the wiring and is constant for our optimization problem. The second summation represents the power consumed while charging or discharging the capacitances of the transistor gates and is a linear expression in the speed factors.

To obtain the average switching frequencies, one must either perform logic simulations with appropriate input vectors or apply statistical methods, as in [10] and [19].

E. LP Formulation of the Gate Sizing Problem

The linear program is now composed as follows: First, we define T_i to be the schedule time, τ_i the delay, and S_i the speed constant of gate i. For the primary inputs T is the arrival time of the signal. The total delay of the circuit is

$$T_{\max} = \max_{i \in primaryoutputs} T_i. \quad (9)$$

Now, for every gate in the circuit the following constraints are defined.

1) The n linearized delay models [a PL implementation of (5)]

$$\begin{aligned} \tau_{gate} &\geq c_{1,1} - c_{1,2} S_{gate} + c_{1,3} \sum_i S_i C_{in,i} \\ &, \ldots, \\ \tau_{gate} &\geq c_{n,1} - c_{n,2} S_{gate} + c_{n,3} \sum_i S_i C_{in,i} \end{aligned} \quad (10)$$

2) S is limited

$$S_{\min} \leq S_{gate} \leq S_{\max}. \quad (11)$$

3) Definitions of schedule times [implement (6)]

$$\forall_{j \in fanin(gate)} T_{gate} \geq T_j + \tau_{gate}. \quad (12)$$

4) Definitions for maximum schedule time of circuit [implement (9)]. If the gate is a primary output

$$T_{\max} \geq T_{gate}. \quad (13)$$

5) The objective function is a linear combination of A, P, and $T_{\max}$

$$c_A A + c_P P + c_T T_{\max}. \quad (14)$$

If the circuit has a Boolean network representation with V vertices (one for every gate) and E edges (one for every connection), we can calculate the LP problem size. The number of constraints is $|E| + (2+n)|V| + 4$ (1 for every predecessor relation, 2 per vertex for limitation of S_{gate}, n per vertex for the piecewise linear delay model, 1 to express the total delay $T_{\max}$, 1 to express the total active area A, 1 to express the total power consumption P and 1 more to limit $T_{\max}$). The number of variables is $3|V| + 3$ (per vertex S, τ, and T, and for the global network $T_{\max}$, A, and P).

Because in practical Boolean networks both the fanin and the fanout of a gate are limited, implying $|E| < c|V|$ for some constant $c > 1$, the size of the LP problem is effectively linear in the number of gates in the circuit.

III. THE PL SIMULATOR PLATO

The simulator PLATO [5]–[8] is a piecewise linear simulator, primarily intended for simulating electrical and logical circuits. The component relations are described by a matrix, relating linear, dynamic and complementary variables and equations. The complementary variables and equations together form a Linear Complementarity Problem (LCP), which is the following problem: given a matrix M and a vector $\mathbf{q}$, determine vectors $\mathbf{w}$ and $\mathbf{z}$ satisfying

$$\begin{cases} \mathbf{w} = M\mathbf{z} + \mathbf{q} \\ \mathbf{w} \geq 0, \quad \mathbf{z} \geq 0, \quad \mathbf{w}^T \mathbf{z} = 0. \end{cases} \quad (15)$$

The equations like $\mathbf{w} \geq 0$ are considered component-wise, i.e., $\forall_i w_i \geq 0$. The complementary variables and equations model the piecewise linear behavior of the components. Due to this piecewise linear modeling, electrical, logical and macro models can be used in one circuit description and simulation run. To support these different models, two types of connections (nets) are available: electrical, with voltage and current variables satisfying the Kirchhoff relations, and signal, which have only a voltage-like variable. The connections of a component to the nets, called terminals, also have one of these types. All kinds of components can be used by the simulator: it has no built-in models but uses a mixture of user-supplied models and library models. To solve the system efficiently, the following methods are repeatedly employed:

- The linear equations, determining the values of voltages, currents and signals, are separated from the component

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

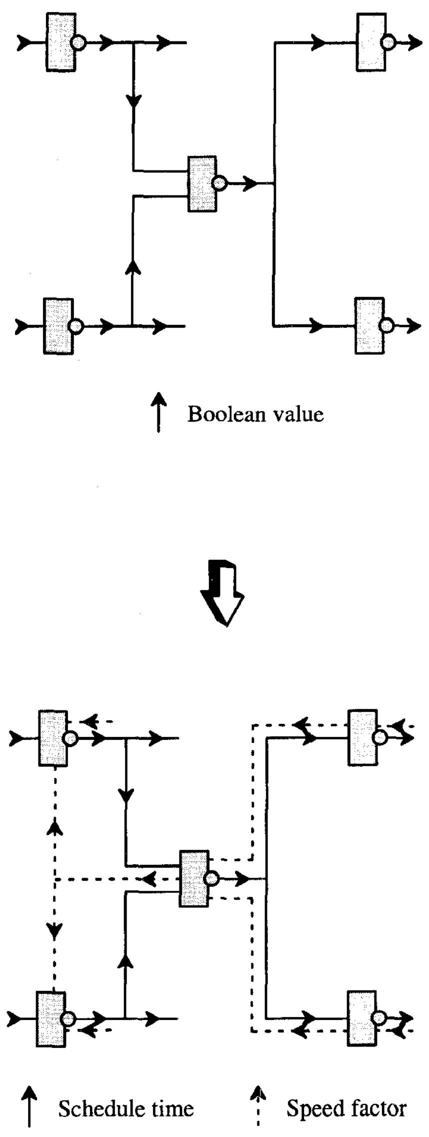


Fig. 2. Circuit to perform optimization in the simulator.

description. These equations are solved with an LU decomposition [12]. Because of the sparse nature of the linear equations (connection matrix), a sparse data structure is used. If the linear equations change, the update of the LU matrices is calculated efficiently with an algorithm that visits only those elements of the matrices that change.

- Solving the LCP is in general, for unrestricted matrices, an NP-hard problem. The Lemke algorithm [16] is a direct, exact method to solve the LCP. It thus has worst case exponential complexity, but in practice almost always finds a solution in very few steps. The solution is found by computing a path through space, performing pivot transformations on the matrix M along its way. Such a pivot effectively exchanges the position of a component of $\mathbf{w}$ with a component of $\mathbf{z}$ in (15), while updating M and $\mathbf{q}$. If after the pivot the new $\mathbf{q}$ becomes nonnegative, a solution is found with the new $\mathbf{z}$ equal to zero. The Lemke algorithm in general might perform off-diagonal pivots. The van de Panne algorithm [20] is a modification, which follows the same path through space toward a solution, but postpones off-diagonal pivot operations until a block-diagonal pivot is assembled and subsequently executed. The PLATO circuit simulator internally takes advantage of hierarchical structuring of equations as often found in large circuits. In such structuring, the van de Panne block-diagonal pivots are easier to accomplish than the single off-diagonal pivots as asked for by the original Lemke algorithm.
- The dynamic equations, a set of linear differential equations, are solved with an integration method. Which method is employed, depends on the problem; usually an implicit linear multistep method is used [13]. To exploit the sparsity and latency of most circuits, the integration method is employed in a multirate scheme, where the circuit is divided dynamically in clusters, and the components in one cluster have the same integration step size, differing from the step size in other clusters. However, if the solution is linear in time, the much simpler and more efficient Forward Euler integration method is used.

The simulator follows a path in the complex space of linear, dynamic and LCP variables. The starting point is determined first by applying the Van de Panne algorithm. The LCP variables and equations determine (convex) regions in the space of linear and dynamic variables. So the state of the LCP, the zero-nonzero partition of the $\mathbf{w}$ and $\mathbf{z}$ vectors, remains valid for some time during the integration. If an entry in either vector becomes negative during the integration, the Van de Panne algorithm is started to change the state of the LCP. The path that the simulator follows can be pictured as a piecewise continuous path (in time) through the space, mixed with (possibly discontinuous) steps. The continuous path is driven by the integration in time, while the (discontinuous) steps are governed by the Van de Panne algorithm.

IV. Modeling the Gate Sizing Problem in the PL Simulator

To obtain the Area-Delay tradeoff curve, the LP problem will be solved for a constraint $T_{\max} \leq f(t)$ with $f(t)$ continuous. The function $f(t)$ may be nonmonotone, but it is simpler to choose a function $f(t) = a - b \times t$, with $b > 0$ and a larger than $T_{\max}(0)$, the value $T_{\max}$ assumes for $\sum S$ minimal [the value $T_{\max}(0)$ can be determined easily from inspection of the circuit]. Then the solution at $t = 0$ is feasible, and a list of solutions for different values of $T_{\max}$ is generated, until no solution can be found. This dynamic problem cannot be integrated easily into the used LP solver. However, the problem can be transformed into an LCP, as will be shown in the next paragraph. This LCP, together with the dynamic behavior, is solved by the simulator PLATO.

An LP problem is converted into an LCP in the following way. Let the LP problem be: find $\min_{A\mathbf{z}_1 \leq \mathbf{b}}\{\mathbf{p}^T\mathbf{z}_1\}$. Apply

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

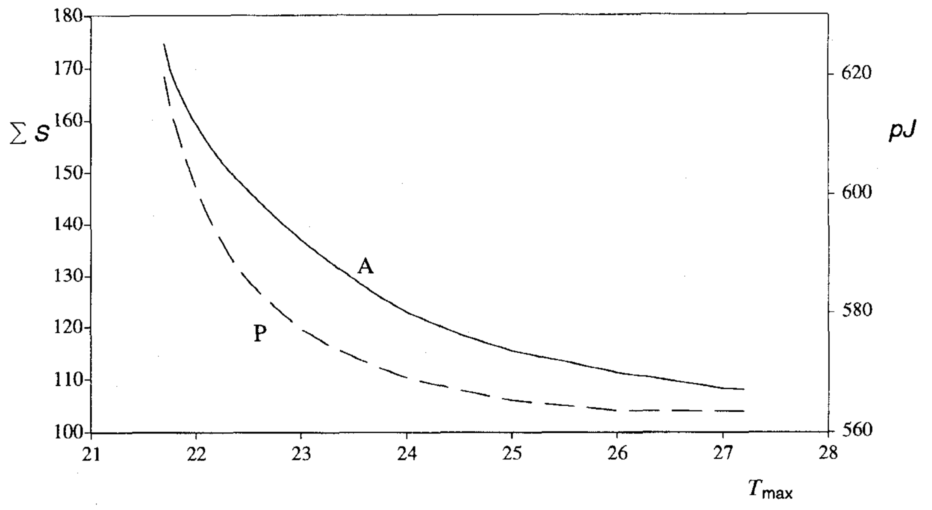


Fig. 3. Area-time and power-time tradeoff curves for the circuit apex2.

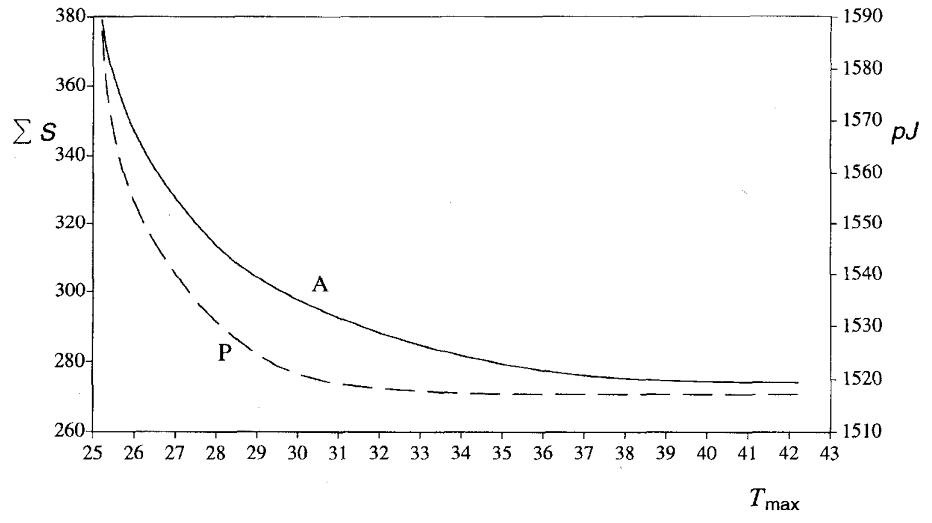


Fig. 4. Area-time and power-time tradeoff curves for the circuit duke2.

the (Karush–)Kuhn–Tucker relations to find the system:

$$\left\{\begin{array}{l}\begin{pmatrix}\mathbf{w}_1\\ \mathbf{w}_2\end{pmatrix} = \begin{bmatrix}0 & A^T\\ -A & 0\end{bmatrix}\begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix} + \begin{pmatrix}\mathbf{p}\\ \mathbf{b}\end{pmatrix}\\ \begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix} \geq 0, \quad \begin{pmatrix}\mathbf{w}_1\\ \mathbf{w}_2\end{pmatrix} \geq 0, \quad \begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix}^T \begin{pmatrix}\mathbf{w}_1\\ \mathbf{w}_2\end{pmatrix} = 0.\end{array}\right. \tag{16}$$

If a solution to this problem exists, the vector $\mathbf{z}_1$ is the solution of the original LP. For an LCP with a matrix as given in (16), the Van de Panne algorithm will always find a solution if it exists. This problem could be entered in a straightforward way into the circuit simulator by creating one large component with this matrix. But the problem can also be represented as the original circuit, with some additional special components. In this way, the sparsity of the network can be employed and the simulation will take much less computer time and resources. In the next section, the conversion of the LP as given in the (10)–(14) into a circuit with appropriate components is discussed.

A. Converting an LP into an LCP Circuit

The LP problem of (10)–(14) is based on the subdivision of the logic circuit into connected components. To use the sparsity of the interconnections, the related LCP problem is converted into a network. Instead of logic values, the schedule times and speed factors are passed between the components. As will be shown in the next paragraphs, it is simpler to use electrical nets: not only schedule times or speed factors are passed over a net, but also extra information needed to solve the optimization problem. Fig. 2 shows the basic conversion step.

Each component has two output terminals, one with its schedule time as signal/voltage value, the other with its speed factor. Furthermore, the schedule times of the components in the fanin set of a component must be known, so the respective nets each have their own input terminal. For the same reason, the nets related to the speed factors of the components in the fanout set are connected to input terminals. The matrix of one

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

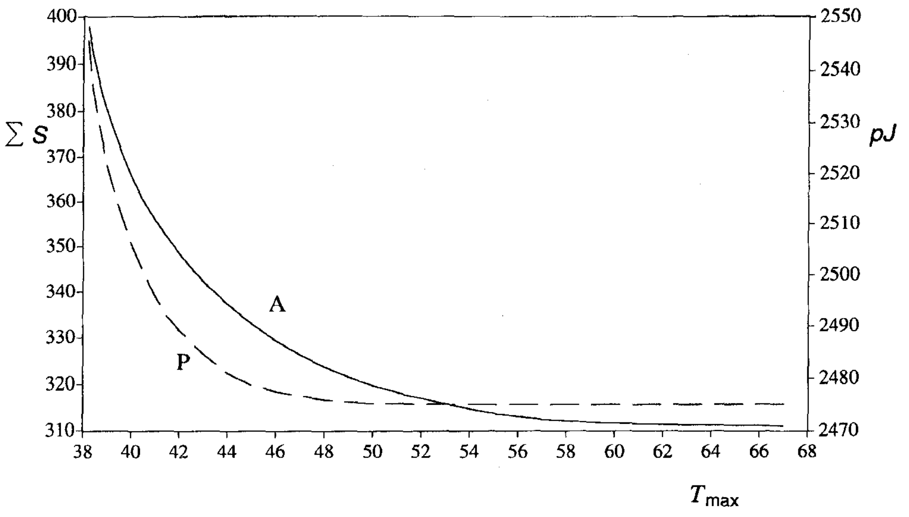


Fig. 5. Area-time and power-time tradeoff curves for the circuit misex3c.

component is constructed from three submatrices, each related to one group of inequalities.

The inequalities $S_{\min} \le S_i \le S_{\max}$ (11), together with the optimization requirement $\sum c_i S_i$ minimal [(7) and (14)], are transformed, according to (16), into the following equations:

$$\begin{cases} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} c_i \\ S_{\max} - S_{\min} \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}. \\ S_i = z_1 + S_{\min} \end{cases} \quad (17)$$

The equations $z_1 \ge 0$ and $w_2 \ge 0$ determine the minimum and maximum bounds of S_i, while the value c_i in the constant vector denotes the value in the objective function.

The second group of inequalities, $T_i = \max\{T_j | j \in fanin(i)\} + \tau_i$ (12), is transformed into system (18). For simplicity, the example gate has only two inputs with delays T_1 and T_2.

$$\begin{cases} \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{pmatrix} T_1 \\ T_2 \\ z_1 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} = \begin{pmatrix} w_{\alpha_1} \\ w_{\alpha_2} \\ w_1 \\ w_3 \\ w_4 \\ w_5 \end{pmatrix}. \\ T_i = z_3 + \tau_i \end{cases} \quad (18)$$

This local matrix is not block-diagonal: the rows related to the variables w_{α_1} and w_{α_2} are not rows of this component's matrix, but the w_3 rows of their respective components. Therefore, the value of z_4, respectively, z_5 must be transported to these components and added to the relevant row. This is done conveniently by setting the schedule time explicitly to an electrical type, with as voltage value the schedule time T_1, and as current value on this connection the value of z_4. Because the currents satisfy the Kirchhoff law, the current on the output T is the sum of these currents of its successors. By adding one entry in the matrix in the row related with w_3, having the value -1 (incoming current!), the correct system is created.

The third group of inequalities, $\tau_i = PL(S_i, S_{out})$ (10), has a form comparable to system (18). Therefore, the nets transferring the speed factors also have a current related to them, and an extra term occurs in the respective w_1 rows. The extra inequality $T_{\max} < f(t)$ will give the same matrix as in (17), but with a right-hand side $[0, f(t)]^T$. This matrix is embedded in a final component that determines the maximal schedule time in the circuit by taking the maximum of the schedule times at the output gates.

The components in the circuit are created according to the equations above, and the connections between them are laid out according to the connections of the original circuit. The extra component for determining and changing the maximum schedule time is connected to the output gates of the circuit. The total number of LCP variables is $2[(4+n)|V|+|E|+2]$, the number of linear (circuit) variables is $4|V| + |E|$. Notice that there is no special component connected to all gates for guiding the optimization process. The gates "know" themselves when their speed factors must change, as is explained in the next section.

B. Finding the LCP Solution

The LCP solution is found by combining the Van de Panne algorithm with an integration in time. By using the inequality $T_{\max} \le a - bt$, the simulation determines the relation between $\sum S$ and the time t. Then it is trivial to find the relation between $T_{\max}$ and $\sum S$. Using a linear function for the time, the numerical integration is as exact as possible and will give no problems. But first an initial solution must be found. This is rather straightforward, when starting from the basic point $\mathbf{z} = 0$. This implies $S_i = S_{\min}$ for all i, and the internal delays τ_i have the values related to these S_i. So only the schedule times T_i and the corresponding LCP variables must be determined. This is done easily by the Van de Panne algorithm. If the schedule time minus the delay of a component is less than the schedule time of one of its predecessors, the corresponding w_k is negative. By performing a block pivot, and due to the form of the equations, the corresponding w_k and z_k both become zero. In network terms, the schedule time is set to the schedule time of the predecessor plus the internal delay.

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

If the initial solution is found, and the forced schedule time at this time point, a, is larger than $T_{\max}$, all currents are zero at the starting point. The other case, that a faster initial solution is sought, will follow from the discussion in the next paragraphs. Because it is a feed-forward network, the calculations are easy and the $T_{\max}$ with $\sum S$ minimal is found.

The dynamic solution starts from this initial solution. With increasing time, the schedule time decreases until no faster solution can be found. The Van de Panne algorithm fails and the simulation stops with an error message. The dynamic solution will follow the piecewise linear tradeoff curve exactly, due to the linear form of the constraint on $T_{\max}$. We will describe the first step of the simulation process, and explain which actions will in general happen in the circuit in the subsequent steps. The first step starts at that time point of the simulation where the inequality in the last component, bounding the schedule time, just becomes invalid, and a new state of the LCP must be found.

The rest of this section is intended to give the reader some insight on the course of the optimization process. It is not mathematically rigorous, but it should give some feeling how we have constructed an "auto-optimizing" circuit.

C. The First Step

Let k be the index of the violated inequality, i.e., $w_k = 0 \wedge \partial w_k/\partial t < 0$. Then the z_k variable can be increased. On circuit level, this means that a current will flow through a schedule time connection, toward the predecessor with the highest schedule time. In this predecessor, two actions happen. First, the current forces the schedule time of this cell to decrease. As in the previous cell, this creates a current through one schedule time connection toward a predecessor. So each cell on the critical path is activated by a current to decrease its schedule time. By the same process, the value of w_1 in each of these cells decreases. As soon as in one cell this value reaches zero, the corresponding z_1 may increase. 'This means that the speed factor of that cell will increase. This decreases the delay of this cell, but increases the delay in its predecessors. However, the total schedule time over this path will decrease, because the global effect is used, i.e., if it would not decrease, w_1 would not decrease. So a new state of the LCP is found that will be valid for the next part of the simulation. The newly found state is translated into the linear equations. These equations can be interpreted by stating that the speed factor in this cell now depends on the time, and that the total schedule time only depends on the time by this relation.

D. Later Steps

This process is repeated at each time point where the current state becomes invalid. This will happen if a speed factor reaches its maximum, if another piece of the piecewise linear approximation is reached, or if another path also becomes critical. In the first case, this cell is replaced by another cell (if it exists) and the curve can be tracked further. In the latter two cases, in general it is necessary to increase not only this speed factor, but also another cell's speed factor. Depending on the size of and the freedom in the circuit, in the latest stages of the simulation many speed factors are manipulated and a large part of all schedule times is changing. Instead of one gate on one critical path, many gates in a critical subnetwork are continuously considered.

Up till now, we have suggested that the speed factors will always increase. However, it may happen (see Fig. 6) that by decreasing one speed factor and increasing another at the same time the total schedule time will decrease. The speed factor may increase again later in the simulation. It is clear, that the schedule time of cells not on the critical path(s) may and will increase, whenever some of their successors become larger and faster.

V. Results

The results can be divided in three parts:

1) the tradeoff curves, the actual results of the simulation run,
2) the comparison with the original LP solver, to compare the results, and
3) the different signal curves, describing the values of voltages and currents as functions of time. These can give detailed information on the optimization process, and can, for example, reveal delay bottlenecks in the circuit.

We have applied both the LP approach from [1] and the new PL simulator approach to the entire set of two-level examples of the MCNC benchmark suite [24], and to a group of other circuits. These other circuits are parameterized versions of a circuit that checks if a given n-bit number is prime, and, if not so, returns the smallest divisor. Example pr$n$ is the n-bit version of this circuit. These prime number circuits have proved to be difficult examples for synthesis and layout software. All circuits were processed by the EUCLID logic synthesis system [2] until a netlist of basic gates was obtained. During this processing, the examples Z5xp1 and Z9sym became equal to 5xp1 and 9sym, respectively. Therefore, Z5xp1 and Z9sym are not listed in the results. All experiments were performed on a HP 9000/750 workstation, running approximately 22 MFLOPS. The speed factor was limited between 1 and 3, and the PL approximation of the $\tau - S$ function had 3 pieces. The constants c_i in (7) and (17) were set to 1.

A. The Tradeoff Curves

The computation of these curves is the main result of this paper. The area-delay and power-delay curves of three bench mark circuits are given in Figs. 3–5. The delay model used is for a 1.5 μ CMOS process. Delays are in nanoseconds. The solid line represents a pure area-delay tradeoff curve, with the area axis drawn on the left hand side. For this test, the area estimate was just $\sum S$. The dashed line represents a pure power-delay tradeoff curve, with the vertical axis on the right hand side. The relative switching frequencies of the gates were obtained by performing logic simulation with out delay before gate sizing, with all input vectors equally likely. In this way, all glitching activity is neglected. Wiring capacitances were estimated with the model from [3]. The relative difference in power consumption for the slowest and

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

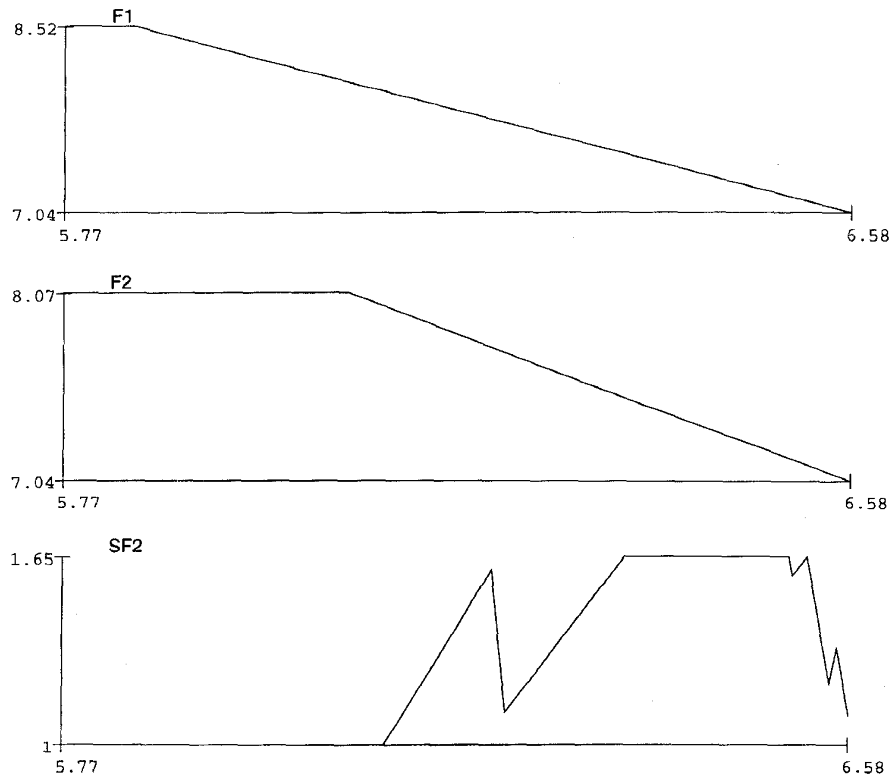


Fig. 6. Schedule times of outputs F1 and F2 of circuit con1, and speed factor of gate F1.

the fastest solutions may seem small, but in these examples most power is consumed while charging and discharging the wires. Apex2 uses 451 pJ in the wiring, duke2 1212 pJ, and misex3c 2043 pJ. As mentioned before, this amount is constant during gate sizing. Other circuits give the same type of tradeoff curves, whose particular shape will depend on the structure of the circuit. The circuits chosen for these figures are so complex that the tradeoff curves seem smooth, but they are still piecewise linear. For simpler circuits, the curves have fewer sections. For more complex circuits, finding the complete tradeoff curve is sometimes difficult, because of numerical problems in finding the left-most part of the curves, i.e., the fastest solutions with minimal $T_{\max}$. These numerical problems are discussed in a later section.

B. Comparison with the LP Solver

The solution of the dynamic LCP has to be compared with the LP solver with respect to the values of the solution, the run times and convergence properties. The results are presented in Table I. For both methods, the fastest solution found is given with its $\sum S$. For the LCP method, the run time for determining the complete tradeoff curve is given, for the LP method only the run time for determining the fastest solution found is given. Furthermore, the number of gates and the number of linear and LCP variables in the simulator is given. The last two columns show the gain in speed and cost in extra area between the slowest (minimal $\sum S$) and fastest solution, determined from the results of the LCP. A % faster value of 55% means that the circuit after gate sizing has a delay of $(100 - 55) = 45\%$ of the original. No circuit can become more than 66.7% faster when $S_{\max}$ is 3. The values in bold face indicate solutions where both methods agree about the fastest solution.

Several aspects must be noted with respect to the results. The problem of finding the fastest solution becomes numerically less stable for larger circuits. Therefore, the LP solver could not always find this solution. In those cases the fastest solution which did converge is tabulated in Table I. For a few cases, the solution with minimal $\sum S$ could not be found with the LP solver, because the right-most part of the tradeoff curve was too flat. This can probably be repaired by a more careful choice of constants in the objective function of the minimization problem. The simulator has comparable problems with the same circuits. For most circuits, the curve could be traced further (sometimes until the end) by changing the values of some internal numerical control parameters.

Table I indicates that both methods agree for most circuits on the results. Especially for the smaller circuits there is no doubt that both methods are equivalent, and only differ by small numerical errors. For the larger examples, it is not always clear which method is better. Because in PLATO the control of numerical errors is more carefully designed than in the LP solver, we suppose that PLATO gives slightly more accurate results.

The run times of both programs are comparable, although the LP solver calculates only one point of the total solution space. In Fig. 7 the run times are plotted against the size of the problem (in gates). This figure suggests that the run

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

TABLE I
RESULTS, RUN TIMES, CHARACTERISTIC SIZES AND GAIN/COST FOR SOLVING LP AND LCP PROBLEM

	LP			LCP			#	#	#	%	%
Name	T_{max}	$\sum S$	cpu(s)	T_{max}	$\sum S$	cpu(s)	gates	linear vars	LCP vars	faster	more S
5xpl[1]	24.98	195.69	4.1	24.98	195.80	2.7	146	721	1285	**29.6**	34.1
9sym[1]	20.56	122.25	1.1	20.46	123.16	1.9	82	400	830	**17.6**	50.2
alu4[1]	79.61	673.97	126.7	79.37	674.75	54.2	589	3722	287	**56.9**	14.6
apex1[1]	91.87	1155.67	220.6	86.42	1232.47	281.0	1103	7639	11126	**44.0**	11.7
apex2	30.91	351.28	20.5	30.91	351.95	18.3	253	1488	2474	**29.6**	39.1
apex3[2,3,4]	419.88	2147.18	470.8	429.58	2138.59	182.9	2131	10722	20934	64.5	0.4
apex4[2,3,4]	1350.00	3648.28	1229.3	1321.49	3662.00	639.9	3622	18551	35911	66.5	1.1
apex5[1,2]	34.69	501.62	25.2	31.22	608.81	166.4	437	2685	4289	44.3	39.3
b12	8.69	69.67	0.5	8.70	70.58	0.9	46	250	444	**22.0**	53.4
bw	11.66	111.69	1.9	11.66	111.74	1.5	89	522	934	**26.2**	25.6
clip	18.03	130.70	2.1	18.03	131.17	3.1	90	554	915	**26.9**	45.7
con1	7.04	25.95	0.1	7.04	27.31	0.2	12	62	118	**17.4**	127.6
cordic	15.70	104.92	0.6	15.70	104.91	0.9	61	316	565	**18.8**	72.0
cps	47.00	747.54	70.3	47.00	750.38	144.6	622	4281	6171	**53.9**	20.6
duke2	25.60	283.89	12.5	25.60	284.11	10.3	224	1363	2209	**36.4**	26.8
e64	35.62	293.31	8.6	35.62	293.74	6.7	239	1330	2174	**31.2**	22.9
ex1010[2,3]	96.49	1140.04	234.9	89.75	1171.36	273.7	1076	7203	10424	38.7	8.9
ex4	21.66	338.07	12.2	21.66	338.58	6.3	270	1477	2669	**28.5**	25.4
ex5	22.23	277.51	16.4	22.23	277.83	11.8	233	1545	2280	**41.4**	19.2
inc	13.24	69.68	0.5	13.24	69.83	0.9	52	312	543	**21.8**	34.2
misex1	14.31	57.43	0.3	14.29	55.13	0.6	34	177	336	**22.2**	62.1
misex2[1]	29.83	211.31	5.7	29.83	211.63	3.9	165	940	1543	**27.1**	28.2
misex3[2,4]	900.00	4785.93	2063.1	882.49	4807.55	736.5	4769	21818	47470	66.7	0.8
misex3c[1]	34.15	441.03	27.2	34.08	441.71	56.1	318	1991	3285	**46.0**	38.9
o64	12.48	427.04	4.5	12.48	427.04	4.9	157	661	1388	**22.6**	172.0
pdc[1,2]	33.63	384.32	11.6	33.44	397.74	35.6	310	1969	3009	**38.6**	28.3
rd53	9.87	27.95	0.2	9.87	28.01	0.3	19	88	210	**13.8**	47.4
rd73	17.31	98.84	1.1	17.31	98.76	1.7	75	452	760	**21.3**	31.6
rd84	20.09	149.76	1.7	20.09	149.85	1.7	131	595	1334	**19.1**	14.3
sao2	14.47	98.99	0.9	14.47	99.43	1.5	74	356	758	**18.8**	34.3
seq[1,2]	86.75	953.32	111.3	74.03	1045.37	200.6	894	6021	8878	56.9	16.9
spla[2]	28.69	370.12	11.7	28.61	374.23	40.8	290	1870	2844	**35.0**	29.0
t481	13.97	59.18	0.3	13.97	59.62	0.4	34	181	315	**18.0**	75.3
vg2	16.77	159.50	2.2	16.77	160.25	4.3	111	638	1093	**28.6**	44.3
xor5	14.62	26.62	0.2	14.62	26.68	0.2	15	95	168	**16.5**	77.8
pr8	25.02	190.76	6.1	25.02	191.04	6.9	151	953	1590	**37.3**	26.5
pr9[2,3]	36.00	555.83	37.6	33.55	631.13	117.3	464	2849	4377	45.3	36.0
pr10[2,3]	86.00	847.14	140.0	71.70	921.59	156.0	799	5754	8190	47.9	15.3
pr11[2,3]	123.44	1695.25	910.0	119.09	1715.17	1059.0	1609	11750	16695	52.8	6.5
pr12[2,3]	357.05	3607.67	5380.0	313.37	3647.56	1438.8	3569	26325	37166	49.2	2.2

Notes: 1: LCP found fastest solution by adjusting numerical control parameters
2: LP could not find fastest solution
3: LCP could not find fastest solution
4: LP could not find slowest solution

times are $\mathcal{O}(n^2)$ for this range of problems. For the LP solver this is justified in [3]. The estimation of the order of the run times of the simulator is complex, because all calculations are performed on sparse data structures, so the density of the linear equations and the connectivity in the system, and the number of time steps determine the run time. If the connectivity and density remain bounded, the run time is linear in the number of time steps, and is expected to be $\mathcal{O}(n)$. However, in the later part of the simulation, during the determination of the fastest possible solution, the connectivity and density grow (as will be explained in the next section). This might explain why the run time tends to grow as $\mathcal{O}(n^2)$..

The cost and gain of the fastest solution compared with the initial, slowest solution show a wide range of values. The schedule time may vary between 16–66% faster, while the extra cost in area may differ from 0.4% up to 172%. There is a tendency for small circuits to have low gain at relatively high costs, while for large circuits high gains are obtained at low costs. This can be explained by the fact that in small circuits there is not much freedom, because each path in the Boolean network contains only a few gates. After a few steps of the simulation, most of the network becomes critical, so decreasing the schedule time is only possible by increasing many speed factors simultaneously. Larger circuits usually have one or two long paths, so by increasing only the speed factors on these paths a faster solution can be found.

C. Signal Curves

For each circuit all values as function of the simulation time can be printed. We have chosen to show only three values of

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

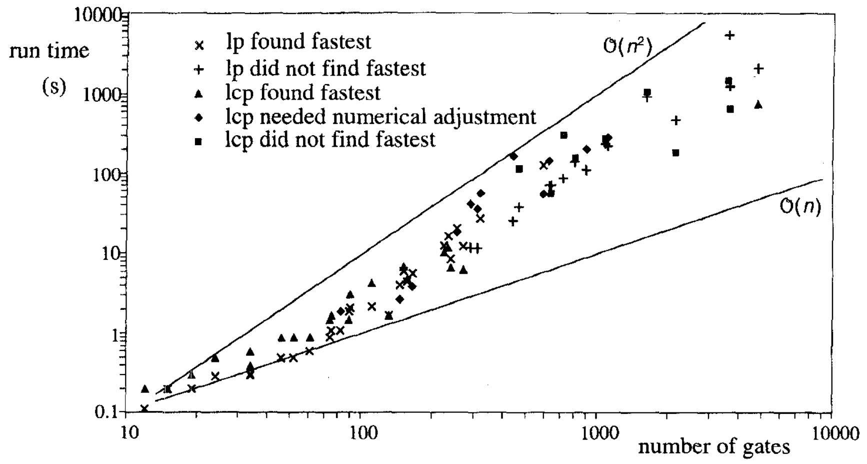


Fig. 7. Run time versus number of gates.

the simplest circuit in the benchmark suite, con1 (Fig. 6). The reason is that many values are trivial (the component's speed factor remains 1) or uninteresting (the currents in the circuit). A third reason is that most values show the behavior that is expected from the model, i.e., the speed factors increase monotonously, and the schedule times show a mixture of increasing and decreasing values, depending on their place in the network. One of the most interesting features, which is many times ignored, is that the speed factors may decrease during the simulation, because it gives room for other speed factors to increase and so decrease the total schedule time of the circuit. This behavior is shown exactly in Fig. 6, where the schedule times of the two outputs of the circuit are shown. First gate F1 is on the critical path, later both gates are on it. The speed factor of gate F2 shows an irregular behavior during the simulation. This can be explained by the fact that this gate has a low internal delay, so decreasing this delay is not so interesting. At certain values for the forcing schedule time, it is therefore advantageous to decrease the speed factor. Fig. 6 shows that for the fastest solution only a small speed factor is found, while for some slower solutions a larger speed factor suits better. Many heuristical approaches to transistor sizing ignore the fact that sizes must sometimes decrease during the optimization process to stay near the optimal solution.

VI. Numerical Aspects

One of the important conclusions that can be drawn from the experiments is that both solution methods for this type of problems show numerical problems for large examples. We will try to analyze these problems in qualitative terms. The characteristics for the problems that the methods can not solve are the same, i.e., both the LP solver and PLATO do not find the fastest solution for (most) problems exceeding a size of about 1000 gates. The symptoms are in many cases the same, namely that a pivot can not be performed because it is too small. Furthermore, the run time increases disproportionately when finding a faster solution, so most time is spent in the left-most part of the tradeoff curve.

This last symptom can be explained by the fact that (in the simulator) the length of the time interval between two subsequent changes of the state of the LCP shortens and the number of matrix entries increases. These facts can be explained by the increasing interdependency between the gates, so more speed factors are changed to decrease the total schedule time.

There may be two reasons for the numerical problems: the matrix of the LP/LCP problem may be ill-conditioned, and/or the convex hull spanned by the inequalities is very flat near the optimal solution. The first case is not likely, as can be seen from the inequalities (10)–(13). The coefficients are all close enough to 1, so an ill-conditioned matrix will occur if two inequalities determine nearly equal hyper-planes. This is not the case. For the LCP the same reasoning shows that both the linear equations and the LCP equations are not ill-conditioned. So it is most likely that the convex hull spanned by the inequalities is very flat near the optimal solution.

VII. Conclusion

The approach to compute tradeoff curves for gate sizing with a piecewise linear simulator has proved to be very effective. Entire tradeoff curves can be computed with about as much CPU time as it takes to get one point on the curve with the LP approach. There are some numerical problems with circuits with more than 1000 gates, but it should be noted that for these large circuits still a substantial part of the tradeoff curve is obtained. Because this is from the designers point of view probably the most interesting part (the right part), these partial results are still useful.

The topic of numerical stability will be subject of further research, because we believe that by careful analysis of the source of the numerical problems we might find a way to avoid them.

References

[1] M. R. C. M. Berkelaar and J. A. G. Jess, "Gate sizing in MOS digital circuits with linear programming," in *Proc. Euro. Design Automation Conf. 1990*, pp. 217–221.

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.

[2] M. R. C. M. Berkelaar and J. F. M. Theeuwen, "Real area-power-delay trade-off in the EUCLID logic synthesis system," in *Proc. IEEE Custom Integrat. Circuits Conf. 1990,* pp. 14.3.1–14.3.4.

[3] M. R. C. M. Berkelaar, "Area-power-delay trade-off in logic synthesis," Ph.D. dissertation, Eindhoven Univ. Technol., Eindhoven, The Netherlands, 1992.

[4] R. K. Brayton, R. Rudell, A. L. Sangiovanni-Vincentelli, and A. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. Computer-Aided Design,* vol. CAD-6, pp. 1062–1081, Nov. 1987.

[5] H. W. Buurman, "From circuit to signal: Development of a piecewise linear simulator," Ph.D. dissertation, Eindhoven Univ. Technol., Eindhoven, The Netherlands, 1993.

[6] J. T. J. van Eijndhoven and M. T. van Stiphout, "Latency exploitation in circuit simulation by sparse matrix techniques," in *Proc. Int. Symp. Circuits Syst.,* Espoo, Finland, June 7–9, 1988.

[7] J. T. J. van Eijndhoven, M. T. van Stiphout, and H. W. Buurman, "Multirate integration in a direct simulation method," in *Proc. Euro. Design Automation Conf., (EDAC),* Glasgow, Scotland, Mar. 12–15, 1990, pp. 306–309.

[8] J. T. J. van Eijndhoven, "Piecewise linear analysis," in *Analog Circuits: Computer Aided Analysis and Diagnosis,* T. Ozawa, Ed. New York: Marcel Dekker, Mar. 1988, pp. 65–92.

[9] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proc. IEEE Int. Conf. Computer-Aided Design,* 1985, pp. 326–328.

[10] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proc. 29th ACM/IEEE Design Automation Conf.,* 1992, pp. 253–259.

[11] L. A. Glasser and L. P. J. Hoyte, "Delay and power optimization in VLSI circuits," in *Proc. IEEE Design Automation Conf.,* 1984, pp. 529–535.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations,* 2nd ed. Baltimore: Johns Hopkins Univ. Press, 1989.

[13] E. Hairer, S. P. Noersett, and G. Warner, *Solving Ordinary Differential Equations I.* Berlin: Springer-Verlag, 1987.

[14] K. S. Hedlund, "Models and algorithms for transistor sizing in MOS circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design,* 1984, pp. 12–14.

[15] W. H. Kao, "Algorithms for automatic transistor sizing in CMOS digital circuits," in *Proc. 22nd Design Automation Conf.,* 1985, pp. 781–784.

[16] C. E. Lemke, "On complementary pivot theory," in *Mathematics of the Decision Sciences—Part 1,* G. B. Dantzig and A. F. Veinott, Eds. Providence, RI: American Math. Soc., 1968, pp. 95–114.

[17] D. Marple, "Transistor size optimization in the tailor layout system," in *Proc. IEEE Design Automation Conf.,*, 1989, pp. 43–48.

[18] M. D. Matson, "Optimization of digital MOS VLSI circuits," in *Proc. Chapel Hill Conf. VLSI,* 1985, pp. 109–126.

[19] F. N. Najm, "Transition density, a stochastic measure of activity in digital circuits," in *Proc. 28th ACM/IEEE Design Automation Conf.,* 1991, pp. 644–649.

[20] C. van de Panne, "A complementary variant of Lemke's method for the linear complementarity problem," *Math. Programming,* vol. 7, pp. 283–310, 1974.

[21] A. U. Ruehli, P. K. Wolff, and G. Goertzel, "Power and timing optimization of large digital systems," in *Proc. IEEE Int. Symp. Circuits Syst.,* 1976, pp. 402–405.

[22] S. S. Sapatnekar, V. B. Rao, and P. M. Vaidya, "A convex optimization approach to transistor sizing for CMOS circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design,* 1991, pp. 482–485.

[23] J. Shyu, A. Sangiovanni-Vincentelli, J. P. Fishburn, and A. E. Dunlop, "Optimization-based transistor sizing," *IEEE J. Solid-State Circuits,* vol. 23, pp. 400–409, Apr. 1988.

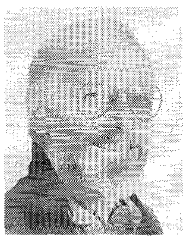
[24] S. Yang, "Logic synthesis and optimization benchmarks user guide version 3.0," Rep. Microelectron. Center of North Carolina, 1991.

Michel R. C. M. Berkelaar was born on September 24, 1959 in Noordwijkerhout, The Netherlands. He received the M.S. degree in electrical engineering "cum laude" from the Eindhoven University of Technology in 1987 and in 1992, he received the Ph.D. degree, which was based on his work on logic synthesis, also from the Eindhoven University of Technology.

In 1987, he joined the Design Automation Section of the Department of Electrical Engineering, Eindhoven University of Technology as a researcher. In 1992, he joined the permanent staff of the Design Automation Section. During 1994–1995, he spent a year as a visiting scientist at the IBM T. J. Watson Research Center.

Pim H. W. Buurman was born on June 4, 1961 in Wageningen, The Netherlands. He studied mathematics at the Eindhoven University of Technology, where he received the M.S. degree in 1987. In 1993 he received the Ph.D. degree, also from the Eindhoven University of Technology, on the design of the piecewise linear simulator PLATO.

He worked in the Design Automation Section of the Eindhoven University of Technology from 1988 to 1993. He is currently with Xirion BV, Software Consultancy, De Meern, The Netherlands.



Jochen A. G. Jess was born on April 13, 1935 in Dortmund, Germany. He received the M.S. degree from the Rheinisch-Westfalische Technische Hochschule Aachen, Germany, 1960 and the Ph.D. degree from the Aachen University of Technology, Germany in 1963.

From 1963 to 1968, he was a research staff member at the Institut fur Nachrichtensysteme, Karlsruhe University of Technology. From 1968 to 1969, he was a Visiting Professor with the Department of Electrical Engineering, University of Maryland, College Park. From 1969 to 1971, he was a senior staff member at the Karlruhe University of Technology. Since 1971, he has been Professor and Head of the Design Automation Section at the department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. His current interests are in the design and design automation of integrated circuits, in particular layout design, logic design, design of architectures, and formal verification. He is a coauthor of more than 75 papers.

Dr. Jess is a member of the Board of the European Design and (Design) Automation Association (EDAA). He has served as a program and general chair for ICCAD93 and ICCAD94.

Authorized licensed use limited to: Eindhoven University of Technology. Downloaded on July 07,2010 at 10:02:09 UTC from IEEE Xplore. Restrictions apply.