Maximum Mean Weight Cycle in a Digraph and Minimizing Cycle Time of a Logic Chip

Christoph Albrecht Bernhard Korte Jürgen Schietke Jens Vygen

Research Institute for Discrete Mathematics, University of Bonn
Lennéstr. 2, 53113 Bonn, Germany

Abstract

The maximum mean weight cycle problem is well-known: given a digraph G with weights $c : E(G) \to \mathbb{R}$, find a directed circuit in G whose mean weight is maximum. Closely related is the minimum balance problem: Find a potential $\pi : V(G) \to \mathbb{R}$ such that the numbers $slack(e) := \pi(w) - \pi(v) - c((v,w))$ $(e = (v,w) \in E(G))$ are optimally balanced: for any subset of vertices, the minimum slack on an entering edge should equal the minimum slack on a leaving edge. Both problems can be solved by a parametric shortest path algorithm.

We describe an application of these problems to the design of logic chips. In the simplest model, optimizing the clock schedule of a chip to minimize the cycle time is equivalent to a maximum mean weight cycle problem. It is very important to find a solution with well-balanced slacks; this problem, in the simple model, is a minimum balance problem.

However, in practical situations many constraints have to be taken into account. Therefore minimizing the cycle time and finding the optimum slack distribution are more general problems. We show how a parametric shortest path algorithm can be extended to solve these problems efficiently.

Computational results with recent IBM processor chips show that the cycle time reduces considerably. Moreover, the number of critical paths (with small slack) decreases dramatically. As a result we obtain significantly faster chips. The running time of our algorithm is reasonable even for very large designs.

1 Introduction

In this paper we consider the maximum mean cycle problem in digraphs, the minimum balance problem, and generalizations. We show how these problems apply to a major problem in the design of very large scale integrated (VLSI) circuits. We encounter the rare case that the original practical problem (with all constraints and without any simplification!) can be solved optimally by efficient algorithms.

A main goal in the design of logic chips is maximizing its frequency. All computations on a chip are synchronized by certain storage elements which receive a clock

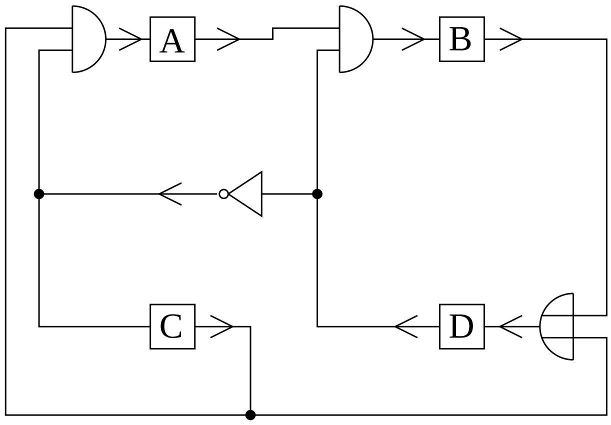

Figure 1: A sample chip

signal periodically. The period is called the cycle time; its inverse is the frequency of the chip.

In each period each storage element (latch) stores one bit. In the next period a fixed logical function is evaluated; the inputs are the bits stored in the previous period and some external inputs. Some of the output bits of the logical function are forwarded to the exterior, the others are stored in the latches; they will be input to the function in the next period.

The begin and end of a period for a certain latch is determined by a periodical clock signal arriving at that latch. The output bit of the function must arrive at this latch before the clock signal, and the propagation of this bit for computation in the next period begins when the clock signal has arrived.

The computation is done by a network of logical gates through which signals are propagated. If the clock signals arrive at all latches simultaneously, then the minimum possible cycle time is determined by the slowest computation, i.e. the longest path in the network. Here the length of a path is its propagation delay; this depends, among other things, on the number and type of gates and their positions. In this paper we assume these lengths to be fixed (already optimized).

Consider the very primitive example of Figure 1: we have four latches (A, B, C, D) and seven paths between pairs of latches, each containing one or two gates. In this example there are no external inputs and outputs. For our purposes, the relevant information is shown in Figure 2: the latches, the paths, and their lengths (which reflect the propagation time along the paths). From these numbers we see that the cycle time must not be shorter than 14 time units: this is the time the signal from D to A needs. So we have simultaneous clock signals at time 0, 14, 28, 42, and so on.

However, if we allow individual clock arrival times at the latches, we can do better: By having clock signal arrival times at latch B one unit earlier and at latch D four units earlier (than at A and C), we can achieve a cycle time of 10:

- Clock signal at latch A: $0, 10, 20, \ldots$

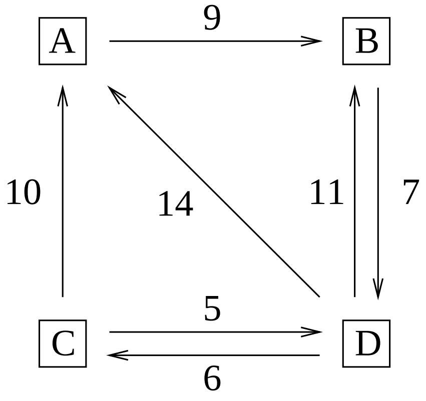

Figure 2: Latch graph of the sample chip

- Clock signal at latch B: $-1, 9, 19, \ldots$
- Clock signal at latch C: $0, 10, 20, \ldots$
- Clock signal at latch D: $-4, 6, 16, \ldots$

One easily checks that all data signals arrive in time: for example the path from D to B has length 11, and the allowed time is $9 - (-4) = 13$ units.

Observe that there exists a circuit in the "latch graph" of Figure 2 which has mean weight 10. Indeed, the maximum mean weight of a circuit equals the optimum cycle time.

So the problem reduces to a maximum mean cycle problem: given a digraph G with edge weights, find a directed circuit in G whose mean weight (total weight divided by number of edges) is maximum. Karp [1978] showed that this problem can be solved in $O(nm)$ time, where n and m denote the number of vertices and edges, respectively. Although this is still the best theoretical bound, other algorithms such as the $O(nm + n^2 \log n)$ parametric shortest path algorithm of Young, Tarjan and Orlin [1991] are faster in practice.

However, the above solution has a serious disadvantage: four out of seven paths have zero slack: if any of these propagation times is larger than estimated, the chip will not function correctly anymore. To make the design more robust we prefer to have as large slacks as possible on as many paths as possible. For example if we change the clock signal arrival times at latch C to $-4, 6, 16, \ldots$, the solution remains valid, but the paths incident to C now all have slack at least 4. We of course prefer such a solution.

There are two ways of formalizing this concept: Let G be a digraph and $c : E(G) \to \mathbb{R}$. First, one could consider the vector of all slacks in nondecreasing order. We prefer one solution to another if this vector is lexicographically greater. In other words, we look for a potential $\pi : V(G) \to \mathbb{R}$ such that the vector of slacks $slack(e) := \pi(w) - \pi(v) - c((v, w))$ $(e = (v, w) \in E(G))$ in nondecreasing order is lexicographically maximum.

The second way of formalizing the concept is to consider the following necessary condition for a solution to be called optimal: For any latch, and in fact for any group of latches, the minimum slack on an entering edge should equal the minimum slack on a leaving edge:

$$\begin{aligned} & \min\{slack(v,w) : v \notin X,\ w \in X\} \\ = \ & \min\{slack(v,w) : v \in X,\ w \notin X\} \quad \text{for all } X \subseteq V(G). \end{aligned} \tag{1}$$

If this were not true, we could improve the solution by changing the arrival time of all latches in a group X violating (1) by some constant.

In fact, the two models are equivalent. If G is strongly connected, (1) is satisfied by a unique $\pi : V(G) \to \mathbb{R}$ (up to addition of a constant); see Schneider and Schneider [1991]. Hence for this π the vector of slacks in nondecreasing order is lexicographically maximum. This problem is known as the minimum balance problem (Schneider and Schneider [1991]; Young, Tarjan and Orlin [1991]).

We shall extend the above observations to a very general model comprising all situations on practical chips. In the general model minimization of the cycle time is not a pure maximum mean cycle problem, but the following more general problem: given a digraph G with edge weights, and a partition of the edge set into red and green edges, find a directed cycle in G maximizing the total weight divided by the number of red edges. This is a special case of the maximum ratio cycle problem (all edge times are 0 or 1), which can be solved in $O(\min\{n^3 \log^2 n, n^3 \log n + mn \log^2 n \log\log n\})$ time (Megiddo [1983]).

Moreover, the slack balancing problem is not a pure minimum balance problem, but the following (more general) problem: given a digraph G with edge weights, and a partition of the edge set into red and green edges, find a potential $\pi : V(G) \to \mathbb{R}$ such that each green edge has nonnegative slack and the vector of slacks of red edges, in nondecreasing order, is lexicographical maximal.

We show that these problems (and extensions discussed in Section 6) can also be solved with a parametric shortest path algorithm in $O(nm + n^2 \log n)$ time.

This paper is organized as follows. After introducing the problem of finding an optimum clocking schedule with a very simple model in Section 2, a general formulation is developed in Section 3. In Section 4 it is shown that the problem reduces to a maximum mean weight cycle problem in a digraph. This can be solved by the algorithm of Young, Tarjan and Orlin [1991], of which an outline is given.

Then we develop an algorithm which takes additional objectives into account. In Section 5 we first consider slacks on signal paths. We obtain a solution where as few as necessary signal paths are critical. In Section 6 it is shown how this algorithm can be modified to increase the slacks on the clocktree paths. Moreover, slacks on signal paths and clocktree paths can be maximized simultaneously.

Computational results with recent IBM processor chips in Section 7 demonstrate the power of our method. The cycle time of the chips is improved by between 2.5 and 5.5 percent. Moreover, the number of critical signal and clocktree paths (with zero or small positive slack) decreases substantially. The running time of the algorithm is reasonable even for latch graphs with several million edges.

2 A simple model

In this section we describe a very simple model for clock scheduling, and show how to optimize the cycle time. This model and the algorithmic solution will be extended in the subsequent sections to reflect very general situations.

A logic chip has a set P of primary (=external) inputs and a set Q of primary outputs. Moreover, there is a set S of storage elements which we assume to store one bit each (e.g. flip-flops or latches). The chip works in time intervals, called cycles, of equal length T. T is called the cycle time of the chip. All the computations of the chip can be described by a boolean function $B : \{0,1\}^{P \cup S} \to \{0,1\}^{Q \cup S}$. In each cycle B is computed, i.e. the values at the primary inputs and the storage elements are used to compute values at the primary outputs and new values to be stored in S; these will be reused as input of B in the next cycle.

The input signals arrive at the primary inputs with the frequency $\frac{1}{T}$, i.e. at time $\xi_p, \xi_p + T, \xi_p + 2T$ etc. for each $p \in P$. The output signals are expected to arrive at the primary outputs at time $\xi_q, \xi_q + T, \xi_q + 2T$ etc. for each $q \in Q$. The numbers ξ_p ($p \in P$) and ξ_q ($q \in Q$) are – together with B – part of the specification of the chip.

The storage elements receive clock signals with the same frequency $\frac{1}{T}$. A special network, the clocktree, distributes one or more clock signals to the storage elements. A simple storage element $s \in S$ works as follows: At certain times $x_s, x_s + T, x_s + 2T$ etc., the result of B for s is expected at the data input of s; at the same time (or slightly later) this value is available at the data output of s and can be used as input for the next computation of B. The time x_s depends on the clock signal arriving at s and is not fixed in advance.

Let G be the graph with $V(G) = P \cup Q \cup S$ whose edges correspond to signal paths: There is an edge (v, w) if the output w of B depends on the input v, i.e. if $B(z)_w \neq B(z')_w$ for some vectors $z, z' \in \{0,1\}^{P \cup S}$ with $z_u = z'_u$ for all $u \in (P \cup S) \setminus \{v\}$. (Subscripts denote components of vectors.)

To be precise, there might be a path from v to w although w does not depend on v. Such so-called false paths can be ignored if they are detected. However, it is often impossible to detect all false paths (this is a *coNP*-hard problem).

For each path from v to w the propagation time of the signal from v to w may vary due to process variations, temperature etc. So we have bounds t_{vw}^{min} and t_{vw}^{max} for the minimum and maximum propagation time over all paths from v to w. Since the network without the latches is acyclic, the graph G and the bounds on the propagation times can be computed by simple forward propagation in topological order.

The chip works correctly if every signal arrives in time:

$$x_v + t_{vw}^{max} \leq x_w + T \qquad \text{for all } (v, w) \in E(G), \tag{2}$$

and no signal arrives too early (i.e. during the previous cycle):

$$x_v + t_{vw}^{min} \geq x_w \qquad \text{for all } (v, w) \in E(G). \tag{3}$$

Here T is the cycle time and x_s ($s \in S$) are variables, while $x_v := \xi_v$ for $v \in P \cup Q$ are given constants. (2) are sometimes called late mode constraints (also known as zero clocking or setup test), (3) are early mode constraints (double clocking or hold test).

In our introductory example (Figure 1 and 2) we ignored early mode constraints. For example, if the minimum propagation delay from latch D to latch C is 3 units (this is quite realistic if the maximum is 6 units), then the clock signal at D must not arrive more than 3 units earlier than the clock signal at C.

In older designs all x_s ($s \in S$) had approximately the same value which was fixed at an early stage. The main advantage of this so-called zero skew approach is its simplicity. A network distributing the clock signals (the clocktree) can be designed in order to guarantee (almost) zero-skew (i.e. simultaneous clock signals); algorithms for this task have been described by Tsay [1993] and Muuss [1994]. However, this is also true for prescribed arrival times with nonzero skew.

The zero-skew approach has several disadvantages. The most striking one is that the cycle time can be improved by choosing the x_s individually. We shall show now how to choose the x_s optimally to minimize the cycle time. Later we show how to obtain a solution where only very few paths on the chip are critical.

In future technologies further disadvantages of the zero-skew approach will also become important. If all storage elements switch at the same time, a very large capacitance has to be loaded simultaneously. The effect is that the supply voltage can fluctuate considerably. Moreover, crosstalk on parallel wires, especially in the clocktree, is a more serious problem with the zero-skew approach. Both effects lead to unpredictable timing behaviour and can cause the chip to fail.

How can we choose the numbers x_s optimally? Let us ignore (3) for a moment. Then minimization of the cycle time T is quite easy: Contract the set $P \cup Q$ in G to a special vertex r. In the resulting graph G' we define edge weights $c((v,w)) := t^{max}_{vw}$ for $(v,w) \in E(G) \cap (S \times S)$ and

$$\begin{aligned} c((r,w)) &:= \max\left\{t^{max}_{pw} + \xi_p : p \in P,\ (p,w) \in E(G)\right\}, \\ c((v,r)) &:= \max\left\{t^{max}_{vq} - \xi_q : q \in Q,\ (v,q) \in E(G)\right\}. \end{aligned}$$

Then we have:

Proposition 2.1 (Vygen [1996]) *The maximum mean weight of a directed circuit in G' with respect to c equals the minimum cycle time T such that values x_s $(s \in S)$ satisfying (2) exist.*

Proof: Let T and x_s ($s \in S$) be a solution of (2), and let C be any circuit in G', say with vertices $v_0, v_1, \ldots, v_k = v_0$ in this order. Setting $x(r) := 0$ we have

$$\sum_{i=1}^{k} c((v_{i-1}, v_i)) \;\le\; \sum_{i=1}^{k} (T + x(v_i) - x(v_{i-1})) \;=\; kT,$$

so the mean weight of C is at most T.

On the other hand, if T is the maximum mean weight of a circuit in G', define $c'(e) := T - c(e)$. There is no negative circuit in G' with respect to c', so we can compute a shortest path potential $\pi : V(G') \to \mathbb{R}$ with $\pi(w) \leq \pi(v) + c'(v,w)$ for all $(v,w) \in E(G')$. Setting $x(v) := \pi(r) - \pi(v)$ for all latches v we obtain a solution satisfying (2). □

So the problem reduces to a maximum mean cycle problem, which is well-solved (Karp [1978], Young, Tarjan, Orlin [1991]).

3 The general problem

To formulate the general problem we have to take a closer look at the storage elements. The clock input of a storage element $s \in S$ has the value 1 in the time from a_s to b_s, then again from $a_s + T$ to $b_s + T$, from $a_s + 2T$ to $b_s + 2T$ and so on. In the remaining time it has the value 0.

When the clock input has the value 1, the storage element is open, i.e. it stores the value currently seen at the data input. When the clock input has value 0, the stored bit remains unchanged. (Sometimes the roles of 0 and 1 are interchanged, but this does not matter.) At any time, the stored bit is available at the data output for subsequent computations.

To model this situation we need two variables per storage element instead of just one. We may shift the clock input for each $s \in S$ by some value y_s, meaning that the clock input has the value 1 in the time intervals $[a_s+y_s, b_s+y_s]$, $[a_s+y_s+T, b_s+y_s+T]$, $[a_s + y_s + 2T, b_s + y_s + 2T]$ and so on. We assume that the length of the interval $b_s - a_s$ cannot be changed.

Since the shifting times y_s have to be realized by a clocktree, it is reasonable to impose a lower bound l_s and an upper bound u_s on y_s for each storage element s:

$$l_s \leq y_s \leq u_s \qquad (s \in S), \tag{4}$$

Moreover, we have a variable x_s for the time when the data signal is valid at the data input of s. Of course, we must have

$$a_s + y_s \leq x_s \leq b_s + y_s \qquad (s \in S), \tag{5}$$

and the data signal should remain valid within the whole interval $[x_s, b_s + y_s]$. For primary inputs and outputs v we set $y_v = 0$ and $a_v = b_v = x_v$ (this value corresponds to ξ_v in the previous section).

A data signal might encounter more than one storage element per cycle (for example in designs using transparent latches). So for each signal path from v to w it has to be specified whether a signal starting at s within the time interval $[a_v, b_v]$ must arrive before b_w (i.e. in the same cycle) or before $b_w + T$ (i.e. in the next cycle). In the first case we set $\zeta_{vw} := 0$, in the second case we set $\zeta_{vw} := 1$.

Then the late mode constraints read as follows:

$$x_v + t_{vw}^{max} \leq x_w + \zeta_{vw} T \qquad \text{for all } (v,w) \in E(G), \tag{6}$$

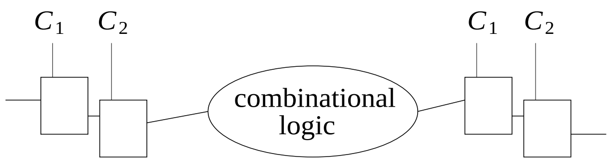

Figure 3: Two master-slave latches with clock signals C_1 and C_2 and a signal path in between

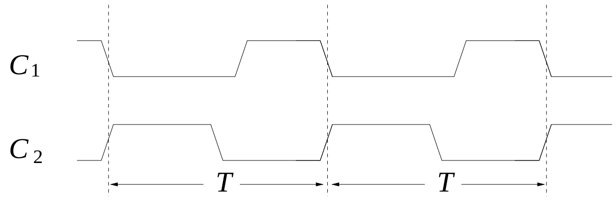

Figure 4: Clock signals at a master-slave latch: voltages as functions of time (0 = low voltage, 1 = high voltage).

where G is defined as in the previous section. For the early mode constraints we have to take the whole intervals into account where the storage elements are open:

$$a_v + y_v + t_{vw}^{min} \geq b_w + y_w + (\zeta_{vw} - 1)T \qquad \text{for all } (v, w) \in E(G). \tag{7}$$

This describes a quite general model. In the simple model of Section 2 we considered only the case where $a_s = b_s$ for all $s \in S$ and $\zeta_{vw} = 1$ for all $(v, w) \in E(G)$. Although it is technically impossible to generate a clock signal which is 1 at a single point of time only, the simple model has some practical relevance: A commonly used storage element is the so-called master-slave latch. Figure 3 shows two master-slave latches with a signal path in between. A master-slave latch has two clock inputs, one data input and one data output. It works as if it would consist of two simple storage elements, where the data output of the first one is connected to the data input of the second one. Moreover, the clock input of the second part is roughly the inverse of the clock signal for the first part (see Figure 4).

As long as the first clock signal C_1 is 1, the value arriving at the data input is stored. If the first clock signal is 0, it does not change anymore. Now if the second clock signal C_2 is 1, the stored bit is visible at the data output, and the new computations begin. If the falling edge of C_1 arrives at the same time as the rising edge of C_2, then such a master-slave latch behaves like a simple latch which is open only at one point of time. However, it is also possible to have two independent clock signals arriving at the latch. This may of course lead to better solutions. Moreover, it is possible to have combinational logic (without storage elements) in between; this common technique is known as cycle stealing (Sakallah, Mudge, Olukotun [1990], Szymanski, Shenoy [1992], Lin, Ludwig, Eng [1992], Lee, Tang, Wong [1996]).

In the above general model a master-slave latch can be represented simply by two simple storage elements s and s'. Then we shall usually have $t_{ss'}^{min} = t_{ss'}^{max} = 0$. This may be used to eliminate the variable $x_{s'}$ (set it to $\max\{x_s, a_{s'} + y_{s'}\}$). However, this increases the number of constraints. It is not a priori clear whether this substitution is computationally favourable or not; indeed this depends on the structure of the design.

In the next section we show how to efficiently solve the above defined linear program, i. e. minimize the cycle time T subject to constraints (4), (5), (6) and (7). After that we shall distribute slacks on signal and clock paths optimally.

Our model is quite flexible. Additional technical constraints can easily be incorporated. For example one can model dynamic circuits by constraints of the same type as above in a straightforward way. Some other constraints (end-of-cycle test, simultaneous clock signals for registers) will be mentioned in Section 7.

Another advantage of this model is that there is no need to distinguish between master-slave latches, simple level-sensitive latches (where $a_s < b_s$; one also speaks of transparent latches) and edge-triggering flip-flops (where $a_s = b_s$). It includes all previous models for cycle time optimization, see Fishburn [1990], Sakallah, Mudge, Olukotun [1990], Szymanski [1992], Shenoy, Brayton [1992], Deokar, Sapatnekar [1994], Neves, Friedman [1996], and Vygen [1996].

4 Computing the optimal cycle time

Observe that each of the inequality constraints (4), (5), (6) and (7) has one or two x- or y-variables and in addition possibly the special variable T. If a constraint has two x- or y-variables, they have opposite sign. To have exactly two variables per inequality we introduce an artificial variable z_0 (corresponding to r in Section 2) which we assume to have value zero. For technical reasons we substitute $\lambda := -T$ and obtain a linear program of the following very special type:

$$\max \ \lambda$$

subject to

$$\begin{aligned} z_i + \ c_{ij} \qquad &\geq \ z_j \qquad \text{for } (i,j) \in E_1, &(8)\\ z_i + \ c_{ij} - \lambda \ &\geq \ z_j \qquad \text{for } (i,j) \in E_2, &(9)\end{aligned}$$

where λ and $z_0, z_1, z_2, ..., z_n$ are variables, the c_{ij} are constants and $E_1, E_2 \subseteq \{0, \dots, n\} \times \{0, \dots, n\}$. Each constraint (4) corresponds to two elements $(i, 0), (0, i)$ of E_1, each constraint (5) corresponds to one element of E_1, and each of the constraints (6) and (7) corresponds to an element of E_1 or E_2, depending on the ζ-constant.

Note that assuming $z_0 = 0$ causes no loss of generality since adding a constant to all variables z_i does not affect feasibility.

Now we translate our linear optimization problem to a network problem. Given the above linear program we construct a directed graph $G' = (V, E)$ as follows: For each variable z_i there is a vertex v_i. For a constraint of type (8) we have a directed edge from vertex v_i to vertex v_j of cost c_{ij}. For a constraint of type (9) we also have

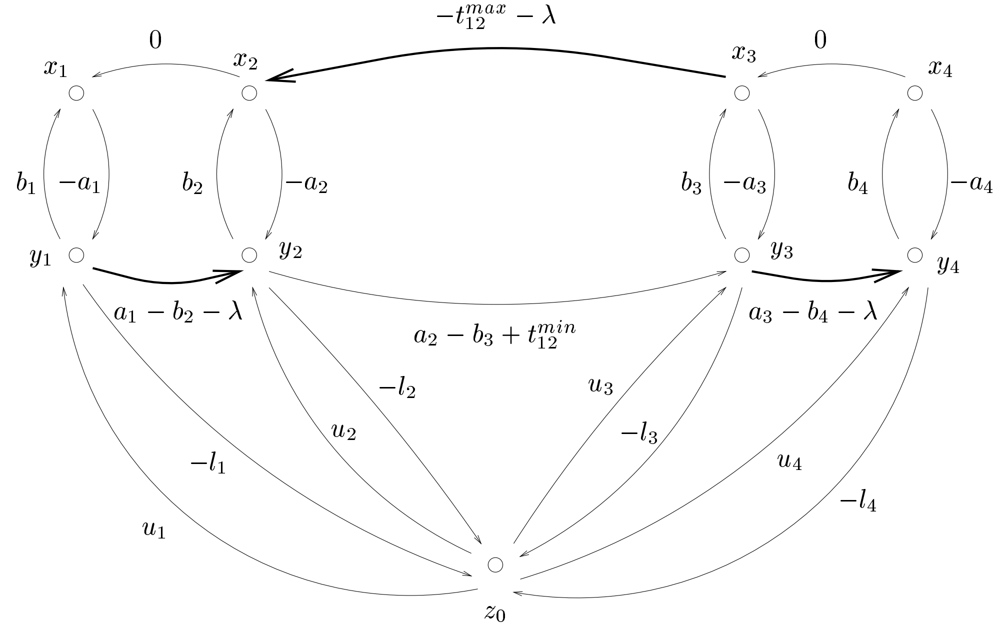

Figure 5: The vertices and edges in G' for two master-slave latches and a signal path in between as shown in Figure 3. The bold edges are parameterized.

an edge from v_i to v_j, but the cost is $c_{ij} - \lambda$. Such an edge is called parameterized: the cost of the edge depends on the parameter λ.

As an example, Figure 5 shows the vertices and edges of G' for two master-slave latches with a signal path in between (from left to right). Each master-slave latch consists of two simple latches, and for each simple latch we have an x-variable (on top) and a y-variable. So there are four variables (vertices) for each master-slave latch; the artificial variable z_0 is represented by the vertex at the bottom. There are eight edges for constraints of type (4), eight for type (5), three edges for type (6) and three for type (7). Three of the edges are parameterized.

We are looking for the maximum value for λ such that values for the variables z_i exist which fulfill all constraints. It is easy to see that such values exist if and only if the digraph G' does not contain a directed circuit of negative cost (negative circuit, for short). This is proved similar to Proposition 2.1; see Szymanski [1992], Shenoy, Brayton [1992], Szymanski, Shenoy [1992], and Deokar, Sapatnekar [1994]. In fact, our problem can be formulated as follows: given a digraph G with edge weights, and a partition of the edge set into red and green edges, find a directed circuit in G minimizing the total weight divided by number of red edges.

The above LP with constraints (9) and (8) might be infeasible in some cases (if there is a negative circuit consisting of unparameterized edges only). Our algorithm detects infeasibility and returns the negative circuit(s) causing the problem. Usually one can cope with this by omitting some early mode constraints: these can be met by inserting buffers (increasing the delay of the path). In fact it is usually not good to take all early mode constraints into account because this might increase the optimum value of the LP (hence the cycle time); one usually prefers inserting a buffer. As we describe in Section 7 we incorporate early mode constraints only after having determined the best possible cycle time. In the following we assume that the LP is feasible.

Previous authors solved this LP either by linear programming (Fishburn [1990], Sakallah, Mudge, Olukotun [1990]) or by binary search with a subroutine testing for a negative circuit (Szymanski [1992], Shenoy, Brayton [1992], Szymanski, Shenoy [1992], Deokar, Sapatnekar [1994], Neves, Friedman [1996]). However, due to its special structure the problem can be solved more efficiently by a direct combinatorial algorithm of Young, Tarjan and Orlin [1991]. We briefly describe their algorithm (which was originally designed for parametric shortest paths) since we shall extend it in Section 5 and 6. For a detailed description and an efficient implementation see also Bünnagel [1998].

The algorithm computes a sequence $-\infty = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_k$ of values for the parameter λ and a sequence $T_1, ..., T_k$ of shortest paths trees in G' from a specified vertex r, the root (in our case we can take the vertex corresponding to the artificial variable z_0: all vertices are reachable from this vertex), such that T_i is a shortest paths tree for all parameters λ with $\lambda_{i-1} \leq \lambda \leq \lambda_i$ ($i = 1, ..., k$). The last value λ_k will be the solution of the linear program.

We start by computing a shortest paths tree for $\lambda = -\sum_{e \in E} |c(e)|$. This value of λ is small enough such that no negative circuit (i.e. directed circuit of negative total cost) exists. The resulting tree is T_1.

Assuming that T_i is already computed we show how to compute λ_i and T_{i+1}. Let P_{rv} be the path from r to v in T_i. We check for each edge $e = (u, v)$ whether the path $P_{ru} + e$ contains more parameterized edges than P_{rv}. If so, $P_{ru} + e$ is a potential pivot path and e is a potential pivot edge. For some value λ_e the path $P_{ru} + e$ will be shorter than P_{rv}.

λ_i is the minimum value λ_e for all potential pivot edges e. One edge $e = (u, v)$ with the minimum value λ_e becomes the pivot edge. We perform a pivot step by deleting the edge with head v from T_i and inserting edge e. The resulting tree is T_{i+1}.

If adding the pivot edge e results in a directed circuit, the algorithm stops. The cost of this directed circuit is zero for λ_e, and $\lambda_e = \lambda_k$ is the maximum value of λ such that G' contains no negative circuit.

The last tree T_k also provides a solution for the variables z_i: one can set z_i to the cost of the path P_{rv_i} in T_k for $\lambda = \lambda_k$. This solution is also called a shortest paths potential.

The worst-case running time of this algorithm (with an efficient implementation) is $O(nm + n^2 \log n)$ where $n = |V(G')|$ and $m = |E(G')|$. However, it is much faster in practice as the experimental results will demonstrate.

We showed how to determine the clocking schedule with the optimum cycle time. However, the solution obtained so far has a serious drawback. Many inequalities of the linear program (in particular all whose corresponding edges belong to the tree T_k) are satisfied with equality. If such a tight inequality corresponds to a signal path, then this path will be critical, i.e. the slack is zero. In the next section we show how the slack can be increased for many critical signal paths. In Section 6 we show how to increase slack on clocktree paths optimally.

We should note that the above method can also be used for static timing analysis with transparent latches, without changing clock arrival times.

5 Balancing slacks on signal paths

Having computed the optimal cycle time T subject to the constraints described in Section 3, we now increase the slack on the signal paths. This is very important: First of all, at the time when the clock schedule is optimized, the propagation delays can only be estimated. Positive slacks on most paths make the chip less sensitive to routing detours, process variations and manufacturing skew. Moreover, if one tries to optimize the cycle time further (e.g. by different logic implementation or placement), only few paths have to be considered.

If ϵ_{vw}^{max} and ϵ_{vw}^{min} are the slacks on the signal path from v to w for late mode and early mode respectively, the constraints (6) and (7) become

$$(6') \qquad x_v + t_{vw}^{max} + \epsilon_{vw}^{max} \;\leq\; x_w + \zeta_{vw} T$$

and

$$(7') \qquad a_v + y_v + t_{vw}^{min} - \epsilon_{vw}^{min} \;\geq\; b_w + y_w + (\zeta_{vw} - 1)T.$$

The task is to maximize the slack variables ϵ_{vw}^{max} and ϵ_{vw}^{min} for as many signal paths as possible such that there is a solution for the linear inequalities (4), (5), (6′) and (7′) for a given cycle time T.

To make this precise we introduce the following partial order relation on the set of all solutions:

Definition 5.1 *Let* $(\epsilon_1, ..., \epsilon_k)$ *and* $(\epsilon'_1, ..., \epsilon'_k)$ *be two vectors (of slack variables). Let* π *be a permutation on* $\{1, ..., k\}$ *such that* $\epsilon_{\pi(1)} \leq \epsilon_{\pi(2)} \leq ... \leq \epsilon_{\pi(k)}$ *and* π' *be a permutation such that* $\epsilon'_{\pi'(1)} \leq \epsilon'_{\pi'(2)} \leq ... \leq \epsilon'_{\pi'(k)}$. *We say that solution* $(\epsilon_1, ..., \epsilon_k)$ *is better than solution* $(\epsilon'_1, ..., \epsilon'_k)$ *if* $(\epsilon_{\pi(1)}, ..., \epsilon_{\pi(k)})$ *is greater than* $(\epsilon'_{\pi'(1)}, ..., \epsilon'_{\pi'(k)})$ *in lexicographic order, i.e. if there exists an* l, $1 \leq l \leq k$, *such that* $\epsilon_{\pi(i)} = \epsilon'_{\pi'(i)}$ *for* $i < l$ *and* $\epsilon_{\pi(l)} > \epsilon_{\pi'(l)}$.

The **slack balancing problem** *consists of finding a solution of* (4), (5), (6′) *and* (7′) *such that the vector of all slack variables is best possible.*

In other words, we look for a solution of (4), (5), (6′) and (7′) such that the vector of all slack variables in nondecreasing order is lexicographically maximal.

The following example illustrates this definition: Suppose we have four signal paths, one solution with late-mode slacks (ϵ_{vw}^{max}) 0, 1, 3, 0 and early-mode slacks (ϵ_{vw}^{min}) 2, 3, 1, 0, and another solution with late-mode slacks 0, 2, 2, 0 and early-mode slacks 5, 2, 1, 3. We sort both solutions with increasing slack regardless of the slack being for late mode or early mode: (0, 0, 0, 1, 1, 2, 3, 3) for the first solution and (0, 0, 1, 2, 2, 2, 3, 5) for the second solution. The second solution is better.

It will be shown that any optimum solution for the slack balancing problem has the same vector of slack variables. We now describe an algorithm which finds this solution. It proceeds as follows:

The slacks of all signal paths are increased simultaneously until they cannot be increased anymore, i.e. some constraints, which form a directed circuit, are already tight. Then we take the subset of all signal paths on which the slack can still be increased and continue to increase the slack on these signal paths.

The same digraph $G' = (V, E)$ as in Section 4 is constructed, but the costs are different. The optimal cycle time T is already computed and should not change, it becomes part of the cost of the respective edges.

The parameterized edges are now those which correspond to constraints with ϵ_{vw}^{max} or ϵ_{vw}^{min} (i.e. ($6'$) and ($7'$)), all other edges are not parameterized. The parameter λ now represents the slack of all signal paths.

We first compute again the maximum value λ such that G' contains no negative circuit, using the parametric shortest path algorithm described in Section 4. This value is zero (if T was the optimal cycle time), and so is the slack of all signal paths for which the corresponding edges belong to the zero cost directed circuit C found by the algorithm. Increasing the parameter λ of any of the parameterized edges on C is impossible, because it would result in a negative circuit. All edges on C lose their parameter, only the parameter of all other edges is increased.

C is contracted, and the costs of the edges leaving and entering C are adjusted: Let z be the vertex to which C is contracted, and let w be the vertex of C nearest to the root r in the last tree T_k computed. For a vertex v of C denote by $c(P_{wv})$ the cost of the path from w to v in T_k for parameter λ_k.

For an edge $e = (v, u)$ leaving C, i.e. v belongs to C but u does not, the corresponding new edge $e' = (z, u)$ after the contraction gets the cost $c(e') = c(e) + c(P_{wv})$. For an edge $e = (u, v)$ entering C, the corresponding new edge gets the cost $c(e') = c(e) - c(P_{wv})$.

The algorithm continues to increase the parameter λ and to change the tree such that it remains a shortest paths tree until the next directed circuit of zero cost is found. The value of the parameter λ at this state is again the slack of the signal paths for which the corresponding edges belong to the directed circuit.

We can now prove that the solution computed by this algorithm is the optimum solution with respect to Definition 5.1:

Theorem 5.2 *The algorithm described above finds an optimum solution to the slack balancing problem. Moreover, any optimum solution for the slack balancing problem has the same vector of slack variables.*

Proof: Whenever the algorithm finds a directed circuit of zero cost, the parameters of all parameterized edges of the circuit have the same value and so all the slacks of the corresponding signal paths are equal. Increasing the slack of one of these signal paths is only possible by decreasing the slack of another signal path, but this would result in a solution which is worse. □

The algorithm presented here is a modification of the algorithm for the minimum balance problem described by Young, Tarjan and Orlin [1991]. For the minimum balance problem all edges of the directed graph G are parameterized. Here only some edges are parameterized, namely those edges which correspond to constraints of signal paths with variables ϵ_{vw}^{max} or ϵ_{vw}^{min}.

This is also the reason why we speak of balancing the slacks: the slacks are increased and distributed "equally" on the signal paths.

With an efficient implementation the worst-case running time of the algorithm described above is $O(nm + n^2 \log n)$.

But note that it is not necessary to run the algorithm to the very end. The algorithm can be stopped at any time, e.g. when a certain value of the parameter λ is reached. Then the slack of the signal paths are only increased up to this value. Since slacks exceeding a certain amount are usually not interesting this option is used in practice; see Section 7.

6 Balancing slacks on clocktree paths

Prescribed arrival times for clock signals at latches cannot be realized since process variations make it impossible to predict the exact arrival times. This is usually taken into account by adding a constant to the cycle time. If one has an interval $[y_s - \epsilon_s, y_s + \epsilon_s]$ of valid clock shifts for each latch s (instead of a fixed number y_s) feasible arrival times are much easier to realize and a smaller cycle time can be achieved (see also Neves and Friedman [1996]).

Clocktrees with prescribed skews can be designed by basically the same algorithm as zero-skew clocktrees. Although this can be done quite efficiently, prescribed skews (zero or not) make detours in the clocktree wiring necessary. If one has intervals for the arrival times of the clock signals one can design clocktrees with significantly smaller wirelength; see e.g. Huang, Kahng and Tsao [1995]. We show how to achieve large intervals for as many latches as possible.

So for each latch s we introduce an additional variable ϵ_s: the clock signal is 1 in the interval $[a_s + y'_s, b_s + y'_s]$ for some $y'_s \in [y_s - \epsilon_s, y_s + \epsilon_s]$. All constraints must be met for all possible values of y'_s within this interval. ϵ_s can be considered as the slack on the clocktree path ending at latch s.

With these additional variables ϵ_s for all latches we can reformulate the linear inequalities:

$$(5'') \qquad a_s + y_s + \epsilon_s \ \leq \ x_s \ \leq \ b_s + y_s - \epsilon_s$$

$$(7'') \qquad a_v + y_v - \epsilon_v + t_{vw}^{min} \ \geq \ b_w + y_w + \epsilon_w + (\zeta_{vw} - 1)T.$$

The problem is solved similarly to the slack balancing problem for signal paths (Section 5), but some modifications are needed since inequality (7″) contains the slack variables ϵ_v and ϵ_w of two different latches v and w which are connected by a signal path.

For each constraint (7″) we introduce an additional variable m_{vw} and split the inequality into two:

$$a_v + y_v - \epsilon_v + t_{vw}^{min} \ \geq \ m_{vw}$$

$$m_{vw} \ \geq \ b_w + y_w + \epsilon_w + (\zeta_{vw} - 1)T$$

Now each inequality contains at most one slack variable ϵ_s, and if it contains the variable ϵ_s, then it contains also the corresponding variable y_s.

As before we construct a digraph G' with a vertex for each variable and an edge for each inequality. An edge is parameterized if and only if the corresponding inequality contains a slack variable ϵ_s. But in contrast to the problem of balancing slacks on signal paths (described in Section 5) we now have several inequalities with the same slack variable ϵ_s.

We use a similar algorithm to that of the previous section. Now it might happen that when a directed zero cost circuit C is found and contracted there is an edge e on C and an edge f not belonging to C with the same slack variable. In this case the value of this slack variable is already determined; the parameter of f must not increase anymore. Edge f (and other edges with the same slack variable) get the cost which they have with the current value of the parameter, they are no longer parameterized. Observe that all edges which lose their parameter have at least one vertex belonging to C, namely the vertex corresponding to the variable y_s.

The cost of these edges are adjusted with the contraction of the directed circuit, hence the running time of the algorithm does not change.

In Section 5 we have balanced the slacks on the signal paths, in this section those on clocktree paths. We finally show that it is also possible to balance the slacks on signal paths and clocktree paths simultaneously. Constraint (7) is substituted by

$$(7''') \qquad a_v + y_v - \epsilon_v + t_{vw}^{min} - \epsilon_{vw}^{min} \ \geq \ b_w + y_w + \epsilon_w + (\zeta_{vw} - 1)T.$$

We look for a solution of constraints (4), (5″), (6′) and (7‴). For each constraint (7‴) we introduce two additional variables m'_{vw} and m''_{vw} and replace the constraint by the following three inequalities:

$$
\begin{aligned}
a_v + y_v - \epsilon_v &\geq m'_{vw} \\
m'_{vw} + t^{min}_{vw} - (\zeta_{vw} - 1)T - \epsilon^{min}_{vw} &\geq m''_{vw} \\
m''_{vw} &\geq b_w + y_w + \epsilon_w.
\end{aligned}
$$

Then we apply the same algorithm as above. We obtain:

Theorem 6.1 *The slack balancing problem for signal and clocktree paths (constraints* (4), (5″), (6′) *and* (7‴)*) has a unique optimum solution which can be computed in time $O(nm + n^2 \log n)$ where n is the number of primary inputs, primary outputs and latches and m is the number of signal paths.*

Proof: It can be derived in the same way as in the proof of Theorem 5.2 that the optimum solution is unique and that the algorithm finds it.

In order to see that the running time is of the given order, observe that the new vertices added for the constraints have only one incoming edge. Such an edge can never be exchanged during the parametric shortest path algorithm unless it is contracted.

By the results of Young, Tarjan and Orlin [1991] it is sufficient to show that the total number of pivot steps is $O(n^2)$. During the algorithm let X be the set of all original vertices in G' (not those resulting from subdividing edges) plus all vertices which have emerged by contraction. For each vertex $v \in X$ consider $\varphi(v) = 5|X| - \rho(v)$, where $\rho(v)$ is the number of parameterized edges on P_{rv}. $\varphi(v)$ is positive and bounded by $O(n)$ as is $|X|$.

At each pivot step there is at least one vertex $v \in X$ for which $\rho(v)$ increases, hence $\sum_{v \in X} \varphi(v)$ strictly decreases.

Finally, we show that $\sum_{v \in X} \varphi(v)$ does not increase due to contraction. If a directed circuit is contracted, then $\rho(v)$ can decrease by at most two (for incoming and leaving edges of the directed circuit) plus three times the number by which $|X|$ decreases (the number of vertices in X on the directed circuit minus one). This is compensated by the term $5|X|$ in $\varphi(v)$. Moreover, for each new vertex z which enters X by the contraction there is at least one vertex v with $\varphi(z) \leq \varphi(v)$ which leaves X.

Hence the expression $\sum_{v \in X} \varphi(v)$ never increases and strictly decreases with each pivot step. Since it is $O(n^2)$ initially and nonnegative throughout, the number of pivot steps is $O(n^2)$. □

7 Computational Results

We have implemented the algorithm in C, all runs are on an IBM RISC System/6000 Model 595. Our algorithm has been applied, among others, to the G3 series of IBM S/390 processor chips (L2 and PU) and the latest follow ups (MBA). For details of the design system see Koehl et al. [1998] and Kick et al. [1997]. Table 1 shows the different chips with target cycle time, number of circuits, nets, pins and primary inputs and outputs (IOs, some of which are bidirectional) and the number of signal paths. See also Figure 10 for a placement of the MBA chip.

Chip	target cycle time (ns)	circuits	nets	pins	primary IOs	latches	signal paths
L2	6.5	87177	103590	339351	928	17032	1173132
PU	6.5	164056	171666	591410	744	17265	2670459
MBA	4.46	394257	402373	1441312	586	40639	1475535

Table 1: Characteristics of three chips

In addition to the constraints described so far further technical restrictions had to be taken into account. For example, for some master-slave latches it is required that the data signal arrives at the latch before the rising time of the clock signal of the slave latch (end-of-cycle test). In this case one has a constraint of the form

$$x_v + t_{vw}^{max} \ \leq \ a_w + y_w + \zeta_{vw} T.$$

Moreover, for certain registers (which can be regarded as sets of latches) simultaneous clock signals are required. In this case only one y-variable for these latches is needed. For implementation reasons we still have separate y-variables and constraints of type $y_v + 0 \leq y_w$ and $y_w + 0 \leq y_v$. This explains the differences in the number of vertices and edges of the graph G' in Table 2 to the numbers which one would expect by Table 1.

Chip	nodes	edges
L2	52999	2103937
PU	68932	5433150
MBA	268153	3637831

Table 2: Size of the digraph G' constructed for balancing slacks on signal paths

Our program consists of two main parts: first the constraints are generated by simple forward propagation for late mode and early mode constraints. During the propagation we store at each circuit the set of all primary inputs and latches from which this circuit can be reached, along with the maximum propagation delay for late mode, resp. the minimum propagation delay for early mode. Table 3 shows the running times for the generation of all constraints for the three different chips. A detailed description of the timing analysis program can be found in Schietke [1999]. The running times for simple propagation (computation of arrival times and slacks only) are shown for comparison.

The second step consists of the main optimization algorithm. Rather than increasing the slacks on late mode and early mode constraints simultaneously, we first increase the slacks on late mode constraints up to a certain value λ^{late} while ignoring all early mode constraints. Then we add all early mode constraints and increase the slack on these constraints up to λ^{early}. For this the late mode constraints are added

Chip	timing	creation
L2	104.11	419.71
PU	178.67	1377.62
MBA	859.25	1006.31

Table 3: Total running time in seconds for evalutaing the timing of the chip by simple propagation (timing) and creating all constraints (creation)

with unparameterized edges, such that the slack of late mode constraints with slack smaller than λ^{late} does not decrease and the slack of all other late mode constraints remains at least λ^{late}. The reason for treating late and early mode constraints differently is that early mode problems can usually be fixed quite efficiently by inserting a buffer (=2 inverters). Finally, we balance the slacks on clocktree paths up to a value λ^{clock} as described in Section 6. Again it is assured that the slacks of late and early mode constraints do not decrease below the value to which they were optimized.

Obviously, the result of the optimization depends on the length of the interval given by l_s and u_s specifying by how much the clock signal arrival time can be shifted from its nominal value. For the L2 and PU designers called for $l_s = -0.4$ns and $u_s = 0.4$ns for most of the latches. However, for 42.8% of the slave latches and 1.4% of the master latches of the L2 we had a prescribed arrival time of the clock signal, i.e. $l_s = u_s = 0.0$ns. On the PU 1.4% of all master latches and 25% of all slave latches had $l_s = u_s = 0.0$ns. On the MBA most of the latches had $l_s = -0.4$ns and $u_s = 0.5$ns, but 1.99% of the master latches and 9.65% of the slave latches had $l_s = u_s = 0.0$ns.

Chip		worst slack	number of signal paths with late mode slack ϵ_{vw}^{max} smaller than							
			< -0.2	< -0.1	< 0.0	< 0.1	< 0.2	< 0.3	< 0.4	< 0.5
L2	before optimization	-0.048	0	0	594	731	740	5781	9541	11938
	after optimization	0.313	0	0	0	0	0	0	0	672
PU	before optimization	-0.103	0	1	143	1384	11349	51578		
	after optimization	0.060	0	0	0	44	1617	44285		
MBA	before optimization	-0.224	5	44	400	2633	9901	21780		
	after optimization	-0.051	0	0	28	89	2283	18768		

Table 4: Improvement of the worst slack and the number of critical signal paths for the three different chips

Table 4 shows the worst slack of all late mode constraints before and after optimization with respect to the target cycle time mentioned in Table 1. This means that the L2 could run with a cycle time of $6.5\text{ns} + 0.048\text{ns} = 6.548\text{ns}$ before the optimization and with a cycle time of $6.5\text{ns} - 0.313\text{ns} = 6.187\text{ns}$ after optimization: the cycle time was improved by 5.5%. Similarly, the cycle time of the PU was improved by 2.5% and the cycle time of the MBA by 3.7%.

Table 4 also shows the number of signal paths with late mode slack smaller than -0.2ns, -0.1ns, ... This demonstrates how dramatically the number of critical paths decreases.

In Table 5 the running times for the algorithm for the three chips for computing the optimal cycle time and for balancing slacks for different scenarios with λ^{late},

Chip	cycle time	$\lambda^{late} = 0.1$	$\lambda^{early} = 0.1$	$\lambda^{clock} = 0.05$	$\lambda^{late} = 0.2$	$\lambda^{early} = 0.2$	$\lambda^{clock} = 0.1$
L2	153.21	4.91	6.93	11.56	4.94	31.87	300.54
PU	19.45	22.35	123.33	569.68	113.68	326.17	13004.25
MBA	20.36	37.63	135.48	671.48	81.34	259.43	92490.36

Chip	$\lambda^{late} = 0.3$	$\lambda^{early} = 0.3$	$\lambda^{late} = 0.4$	$\lambda^{early} = 0.4$	$\lambda^{late} = 0.5$	$\lambda^{early} = 0.5$
L2	5.45	97.48	104.80	634.16	282.21	1608.97
PU	1140.04	1768.60				
MBA	212.60	11011.82				

Table 5: Total running time in seconds for the main optimization routine: for computing the optimal cycle time and for balancing slacks on late and early mode constraints and also clocktree paths up to different values for λ^{late}, λ^{early} and λ^{clock}

λ^{early} and λ^{clock} are shown. Again, all late mode slacks are taken with respect to the target cycle times.

Figure 6 shows a frequency distribution of the slacks of all late mode constraints for the MBA for the case that no optimization is possible (i.e. $l_s = u_s = 0.0$ns for all latches s). The first column shows the different intervals, the second column gives the number of signal paths whose slack is within the interval and the third column gives a graphical representation of this number by a proportional number of stars. For example it can be read from Figure 6 that the MBA without optimization has 101 signal paths with $-0.100\text{ns} \leq \epsilon_{vw}^{max} < -0.050\text{ns}$. Figure 7 shows the corresponding frequency distributions of late mode constraints after optimization for $\lambda = 0.2$ns.

Figures 8 and 9 show the same for all early mode constraints. Increasing the slacks on late mode constraints makes the worst slack of all early mode constraints worse, but nevertheless the total number of early mode constraints with negative slack decreases considerably.

Figure 11 shows the effect of slack balancing at a glance. Each line connects the endpoints of a critical path, with respect to the placement shown in Figure 10. The colours have the following meaning:

- red lines represent signal paths with a negative late mode slack;
- yellow lines represent signal paths with a late mode slack between 0.0 and 0.2ns;
- blue lines represent signal paths with a negative early mode slack;
- green lines represent signal paths with an early mode slack between 0.0 and 0.2ns;

The left-hand side is the situation before optimization, the right-hand side shows that after optimization only very few critical areas remain. Figure 13 is the analogous picture for the PU, with respect to the placement shown in Figure 12.

Figures 14 and 15 show the frequency distribution of the y-variables (shifting times of clock signal arrival times at latches) of all master and slave latches respectively. The values correspond to the achieved slacks in Figures 7 and 9. One can observe that more than 75% of all y-variables are within ± 0.2ns.

```
-------------------------------------------------------------------------------------------------
     slacks_late
     number of values: 1475535
     minimum value:    -0.223992
     one star represents 300 values.

  0.400 -  0.450 | 13830 | **********************************************
  0.350 -  0.400 | 11409 | **************************************
  0.300 -  0.350 |  8635 | ****************************
  0.250 -  0.300 |  6754 | **********************
  0.200 -  0.250 |  5125 | *****************
  0.150 -  0.200 |  5239 | *****************
  0.100 -  0.150 |  2029 | ******
  0.050 -  0.100 |  1535 | *****
 -0.000 -  0.050 |   698 | **
 -0.050 - -0.000 |   255 |
 -0.100 - -0.050 |   101 |
 -0.150 - -0.100 |    37 |
 -0.200 - -0.150 |     2 |
 -0.250 - -0.200 |     5 |
 -0.300 - -0.250 |     0 |
-------------------------------------------------------------------------------------------------
```

Figure 6: Slacks of late mode constraints (ϵ_{vw}^{max}) without optimization for the MBA

```
-------------------------------------------------------------------------------------------------
     slacks_late
     number of values: 1475535
     minimum value:    -0.050953
     one star represents 300 values.

  0.400 -  0.450 | 21015 | **********************************************************************
  0.350 -  0.400 | 17637 | **********************************************************
  0.300 -  0.350 | 14539 | ************************************************
  0.250 -  0.300 | 11467 | **************************************
  0.200 -  0.250 |  5018 | ****************
  0.150 -  0.200 |  2121 | *******
  0.100 -  0.150 |    73 |
  0.050 -  0.100 |    23 |
 -0.000 -  0.050 |    38 |
 -0.050 - -0.000 |    27 |
 -0.100 - -0.050 |     1 |
 -0.150 - -0.100 |     0 |
 -0.200 - -0.150 |     0 |
 -0.250 - -0.200 |     0 |
 -0.300 - -0.250 |     0 |
-------------------------------------------------------------------------------------------------
```

Figure 7: Optimized late mode slacks (ϵ_{vw}^{max}) for the MBA

Finally, in Figures 16 and 17 the result of balancing the slacks on clocktree paths is shown. Even though the slacks on late mode and early mode constraints have already been increased up to 0.2ns, the clock signal for most of the latches still does not have to arrive at exactly the prescribed time.

Acknowledgement

We thank the anonymous referees for comments that helped improving the presentation of this paper.

```
------------------------------------------------------------------------------------------------
     slacks_early
     number of values: 1457513
     minimum value:    -0.021301
     one star represents 300 values.

  0.400 -  0.450 | 94877 | ********************************************************************>>>
  0.350 -  0.400 | 79750 | ********************************************************************>>>
  0.300 -  0.350 | 62978 | ********************************************************************>>>
  0.250 -  0.300 | 51393 | ********************************************************************>>>
  0.200 -  0.250 | 55248 | ********************************************************************>>>
  0.150 -  0.200 | 23346 | ********************************************************************>>>
  0.100 -  0.150 | 13643 | *********************************************
  0.050 -  0.100 |  9235 | ******************************
 -0.000 -  0.050 |  5743 | *******************
 -0.050 - -0.000 |   140 |
 -0.100 - -0.050 |     0 |
 -0.150 - -0.100 |     0 |
 -0.200 - -0.150 |     0 |
 -0.250 - -0.200 |     0 |
 -0.300 - -0.250 |     0 |
------------------------------------------------------------------------------------------------
```

Figure 8: Slacks of early mode constraints (ϵ_{vw}^{min}) without optimization for the MBA

```
------------------------------------------------------------------------------------------------
     slacks_early
     number of values: 1457513
     minimum value:    -0.125193
     one star represents 300 values.

  0.400 -  0.450 | 75226 | ********************************************************************>>>
  0.350 -  0.400 | 64770 | ********************************************************************>>>
  0.300 -  0.350 | 60908 | ********************************************************************>>>
  0.250 -  0.300 | 45244 | ********************************************************************>>>
  0.200 -  0.250 | 20418 | ********************************************************************
  0.150 -  0.200 |   622 | **
  0.100 -  0.150 |   260 |
  0.050 -  0.100 |   344 | *
 -0.000 -  0.050 |   502 | *
 -0.050 - -0.000 |   244 |
 -0.100 - -0.050 |     8 |
 -0.150 - -0.100 |     1 |
 -0.200 - -0.150 |     0 |
 -0.250 - -0.200 |     0 |
 -0.300 - -0.250 |     0 |
------------------------------------------------------------------------------------------------
```

Figure 9: Optimized early mode slacks (ϵ_{vw}^{min}) for the MBA

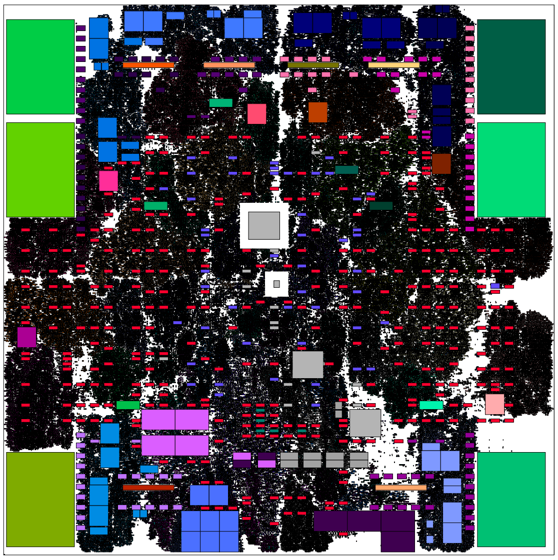

Figure 10: MBA Placement

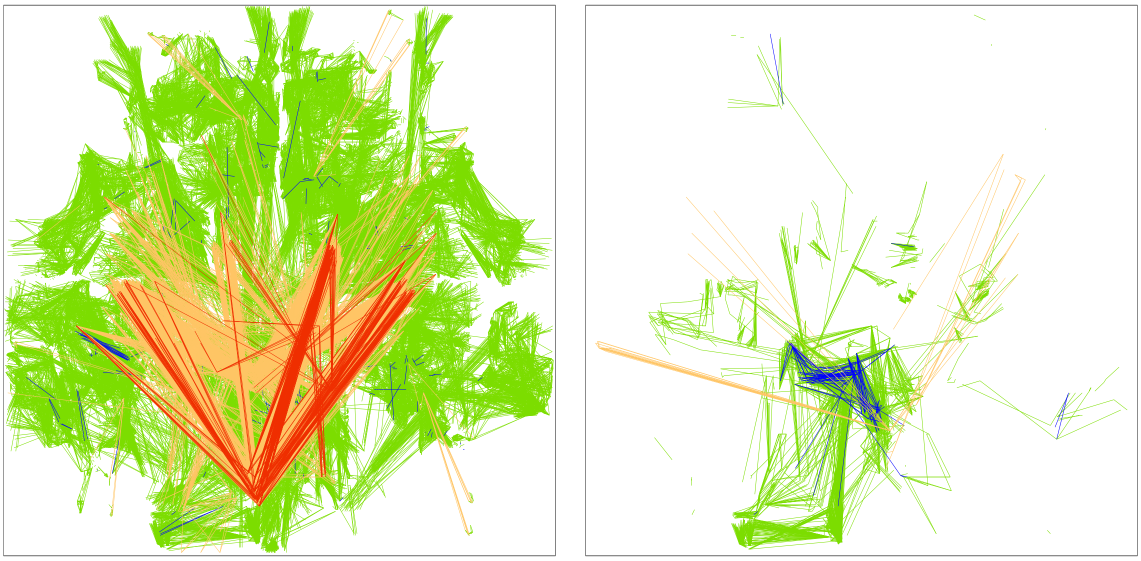

Figure 11: Signal paths on MBA with slack < 0.2ns before and after optimization

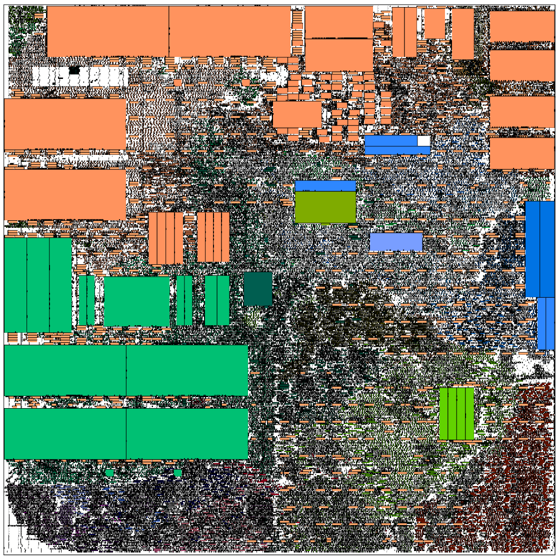

Figure 12: PU Placement

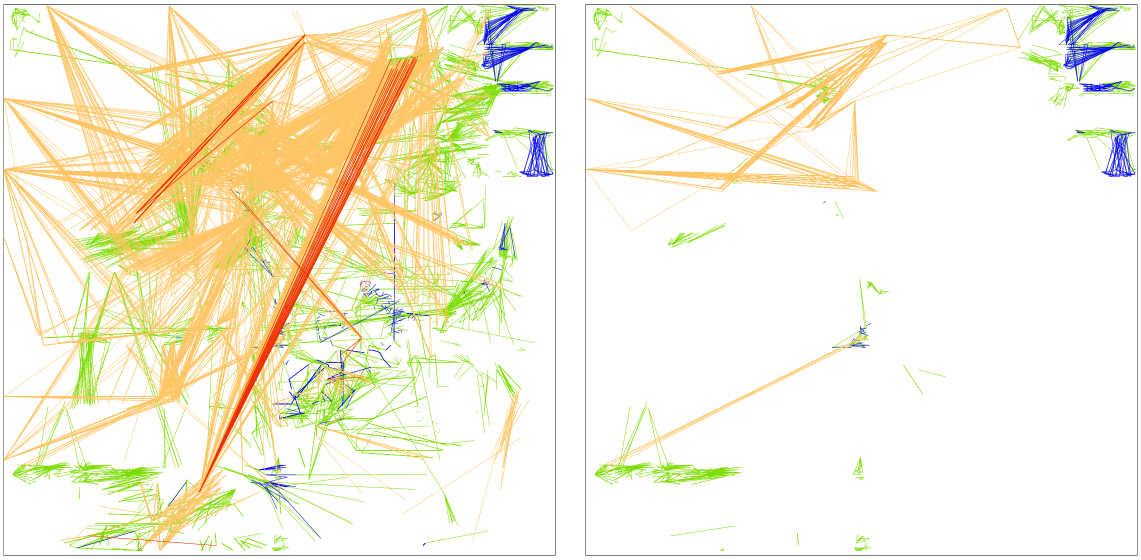

Figure 13: Signal paths on PU with slack < 0.2ns before and after optimization

```
----------------------------------------------------------------------------------------------------
     y_master_shift
     number of values: 31561
     maximum value:     0.500000
     minimum value:    -0.400000
     one star represents 100 values.

  0.500 -  0.550 |  1196 | ***********
  0.450 -  0.500 |    67 |
  0.400 -  0.450 |   214 | **
  0.350 -  0.400 |   381 | ***
  0.300 -  0.350 |   596 | *****
  0.250 -  0.300 |   933 | *********
  0.200 -  0.250 |  1228 | ************
  0.150 -  0.200 |  1544 | ***************
  0.100 -  0.150 |  1534 | ***************
  0.050 -  0.100 |  2312 | ***********************
  0.000 -  0.050 |  3735 | *************************************
 -0.050 -  0.000 |  3911 | ***************************************
 -0.100 - -0.050 |  4411 | ********************************************
 -0.150 - -0.100 |  3290 | ********************************
 -0.200 - -0.150 |  2332 | ***********************
 -0.250 - -0.200 |  1437 | **************
 -0.300 - -0.250 |   804 | ********
 -0.350 - -0.300 |   451 | ****
 -0.400 - -0.350 |  1185 | ***********
----------------------------------------------------------------------------------------------------
```

Figure 14: Shifting times (y) of master latches

```
----------------------------------------------------------------------------------------------------
     y_slave_shift
     number of values: 34237
     minimum value:    -0.400000
     one star represents 100 values.

  0.500 -  0.550 |    19 |
  0.450 -  0.500 |  1077 | **********
  0.400 -  0.450 |   135 | *
  0.350 -  0.400 |   336 | ***
  0.300 -  0.350 |   675 | ******
  0.250 -  0.300 |  1203 | ************
  0.200 -  0.250 |  1922 | *******************
  0.150 -  0.200 |  2742 | ***************************
  0.100 -  0.150 |  3522 | ***********************************
  0.050 -  0.100 |  3716 | *************************************
  0.000 -  0.050 |  6992 | *********************************************************************
 -0.050 -  0.000 |  3728 | *************************************
 -0.100 - -0.050 |  3132 | *******************************
 -0.150 - -0.100 |  1949 | *******************
 -0.200 - -0.150 |   916 | *********
 -0.250 - -0.200 |   444 | ****
 -0.300 - -0.250 |   394 | ***
 -0.350 - -0.300 |   233 | **
 -0.400 - -0.350 |  1102 | ***********
----------------------------------------------------------------------------------------------------
```

Figure 15: Shifting times (y) of slave latches

```
----------------------------------------------------------------------------------------------------
    e_master
    number of values: 31561
    minimum value:    0.000000
    one star represents 150 values.

 0.100 -  0.110 |  7179 | ***********************************************
 0.090 -  0.100 |  1860 | ************
 0.080 -  0.090 |  2110 | **************
 0.070 -  0.080 |  2505 | ****************
 0.060 -  0.070 |  2346 | ***************
 0.050 -  0.060 |  2336 | ***************
 0.040 -  0.050 |  3018 | ********************
 0.030 -  0.040 |  3697 | ************************
 0.020 -  0.030 |  3401 | **********************
 0.010 -  0.020 |  1943 | ************
 0.000 -  0.010 |  1166 | *******
----------------------------------------------------------------------------------------------------
```

Figure 16: Slack on clocktree (ϵ_s) paths of master latches

```
----------------------------------------------------------------------------------------------------
    e_slave
    number of values: 34237
    minimum value:    0.000000
    one star represents 150 values.

 0.100 -  0.110 |  9061 | ************************************************************
 0.090 -  0.100 |  2717 | ******************
 0.080 -  0.090 |  1960 | *************
 0.070 -  0.080 |  2280 | ***************
 0.060 -  0.070 |  2249 | **************
 0.050 -  0.060 |  2189 | **************
 0.040 -  0.050 |  3169 | *********************
 0.030 -  0.040 |  3867 | *************************
 0.020 -  0.030 |  3143 | ********************
 0.010 -  0.020 |  1871 | ************
 0.000 -  0.010 |  1731 | ***********
----------------------------------------------------------------------------------------------------
```

Figure 17: Slack on clocktree (ϵ_s) paths of slave latches

8 References

Bünnagel, U. [1998]: Effiziente Implementierung von Netzwerkalgorithmen. Diploma thesis, University of Bonn, 1998 [in German]

Deokar, R.B., Sapatnekar, S. [1994]: A Graph-theoretic Approach to Clock Skew Optimization. Proceedings of the IEEE International Symposium on Circuits and Systems (1994), 407-410

Doettling, G., Getzlaff, K.J., Leppla, B., Lipponer, W., Pflueger, T., Schlipf, T., Schmunkamp, D., Wille, U. [1997]: S/390 Parallel Enterprise Server Generation3: A balanced system and cache structure. IBM Journal of Research and Development 41 (1997), 405-428

Fishburn, J.P. [1990]: Clock Skew Optimization. Transactions on Computers C-39 (1990), 945-951

Huang, J.H., Kahng, A.B., Tsao, C.-W.A. [1995]: On Bounded-Skew Routing Tree Problem. Proceedings of the 32nd Design Automation Conference (1995), 508-513

Karp, R.M. [1978]: A Characterization of the Minimum Mean Cycle in a Digraph. Discrete Mathematics 23 (1978), 309-311

Kick, B., Baur, U., Koehl, J., Ludwig, T., Pflueger, T. [1997]: Standard-cell-based design methodology for high performance support chips. IBM Journal of Research and Development 41 (1997), 505-514

Koehl, J., Baur, U., Ludwig, T., Kick, B., Pflueger, T. [1998]: A Flat, Timing-Driven Design System for a High-Performance CMOS Processor Chipset. Proceedings of the Conference "Design, Automation and Test in Europe" (1998), 312-320

Lee, J., Tang, D.T., Wong, C.K. [1996]: A Timing Analysis Algorithm for Circuits with Level-Sensitive Latches. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 15 (1996), 535-543

Lin, I., Ludwig, J.A., Eng, K. [1992]: Analyzing Cycle Stealing on Synchronous Circuits with Level-Sensitive Latches. Proceedings of the 29th ACM/IEEE Design Automation Conference (1992), 393-398

Megiddo, N. [1983]: Applying Parallel Computation Algorithms in the Design of Serial Algorithms. Journal of the Association for Computing Machinery 30 (1983), 852-865

Muuss, K. [1994]: Clockskew Optimierung. Diploma thesis, University of Bonn, 1994 [in German]

Neves, J.L., Friedman, E.G., [1996]: Optimal Clock Skew Scheduling Tolerant to Process Variations. Proceedings of the 33rd Design Automation Conference (1996), 623-628

Sakallah, K.A., Mudge, T.N., Olukotun, O.A. [1990]: $checkT_c$ and $minT_c$: Timing Verification and Optimal Clocking of Synchronous Digital Circuits. Proceedings of the IEEE International Conference on Computer-Aided Design (1990), 552-555

Schietke, J. [1996]: Timing-Optimierung beim physikalischen Layout von nicht-hierarchischen Designs hochintegrierter Logikchips. Ph.D. thesis, University of Bonn, 1999 [in German]

Schneider, H., Schneider, M.H. [1991]: Towers and cycle covers for max-balanced graphs. Mathematics of Operations Research 16 (1991), 208-222

Shenoy, N., Brayton, R.K. [1992]: Graph Algorithms for Clock Schedule Optimization. Proceedings of the IEEE International Conference on Computer-Aided Design (1992), 132-136

Szymanski, T.G., [1992]: Computing Optimal Clock Schedules. Proceedings of the 29th ACM/IEEE Design Automation Conference (1992), 399-404

Szymanski, T.G., Shenoy, N. [1992]: Verifying Clock Schedules. Proceedings of the IEEE International Conference on Computer-Aided Design (1992), 124-131

Tsay, R.-S. [1991]: Exact zero skew. IEEE Transactions on Computer-Aided Design 21 (1991), 205-221

Vygen, J. [1996]: Plazierung im VLSI-Design und ein zweidimensionales Zerlegungsproblem. Ph.D. thesis, University of Bonn, 1996 [in German]

Young, N.E., Tarjan, R.E., Orlin, J.B. [1991]: Faster Parametric Shortest Path and Minimum Balance Algorithms. Networks, 21 (1991), 205-221