TOPOLOGICAL SYNTHESIS OF CLOCK TREES FOR VLSI-BASED DSP SYSTEMS

Ivan S. Kourtev and Eby G. Friedman
Department of Electrical Engineering
University of Rochester
Rochester, NY 14627, U.S.A.

Abstract—This paper considers the problem of designing the topology of a clock distribution network for a synchronous digital signal processor so as to satisfy a non-zero clock skew schedule. A methodology and related algorithms for synthesizing the topology of the clock distribution network from a clock schedule derived from circuit timing information are presented. A new formulation of the problem of designing the clock distribution network is given as an efficiently solvable integer linear programming (ILP) problem. The proposed approach is demonstrated on the suite of ISCAS'89 benchmark circuits. Up to a 64% performance improvement is attained on these circuits by exploiting non-zero clock skew throughout the synchronous system. Clock tree topologies that implement the non-zero clock skew schedule based on the synthesis algorithms presented in this paper are described for each of the benchmark circuits.

1 INTRODUCTION

Most advanced digital signal processing (DSP) algorithms are based on the iterative execution of simple operations. Typically, these DSP algorithms are highly regular and can be efficiently *parallelized* and *pipelined* for hardware implementation. This process of pipelining the computation requires the insertion of clocked registers at specific locations throughout the circuit. The vast majority of VLSI/ULSI-based DSP systems are based on a *fully synchronous* synchronization strategy where it is not uncommon for the computational process to depend upon the proper timing of tens of thousands of registers interconnected by hundreds of thousands of logic elements.

For such a synchronous digital systems to function properly, the many thousands of switching events require a strict temporal ordering. This strict ordering is enforced by a global synchronization signal, known as the *clock signal.* For a fully synchronous system to function properly, the clock signal must be delivered to every register at a precise *relative* time. The delivery function is accomplished by a circuit and interconnect structure known as the *clock distribution network* [1].

The nature of the on-chip clock signal has become a primary factor limiting circuit performance, causing the clock distribution network to become a performance bottleneck for high speed VLSI-based DSP circuits. The primary source of the load for the clock signals is shifting from the logic gates to the *interconnect,* thereby changing the physical nature of the load from a lumped capacitance (C) to a distributed *resistive-capacitive* (RC) load [2, 3]. These interconnect impedances degrade the on-chip signal waveform shapes and in-

0-7803-3806-5/97/$10.00 ©1997 IEEE

crease the path delay. Furthermore, statistical variations in the values of the circuit elements along the clock and data signal paths, caused by imperfect control of the manufacturing process and the environment, introduce ambiguity into the signal timing that cannot be neglected. All of these changes have a profound impact on both synchronous design methodologies and circuit performance. Among the most important consequences are increased power dissipation in the clock distribution network as well as increasingly challenging timing constraints that must be satisfied in order to avoid clock hazards [1, 4–7]. The majority of the approaches used to design a clock distribution network target minimal or zero global clock skew [8–10], which can be achieved by different routing strategies [11–14], buffered clock tree synthesis, symmetric n-ary trees [5] (most notably H-trees), or a distributed series of buffers connected as a mesh [1, 4].

This paper addresses the issue of synthesizing the topology of a *non-zero* skew clock distribution network that satisfies the tighter timing constraints required in high performance DSP VLSI complexity systems. First, some background information is reviewed in section 2. The problem and a solution are formulated and the application of the proposed solution to benchmark circuits is described in section 3. Some concluding remarks are offered in section 4.

2 BACKGROUND

In this section, certain important properties of a synchronous digital system are outlined, the model used in this paper to describe these systems is formulated, and the notations and definitions used throughout this paper are introduced. Specifically, in sections 2.1 and 2.2, the fundamental operation of a synchronous system and of *clock scheduling*, respectively, is reviewed. In section 2.3, the tree structure of the clock distribution network is described and analyzed.

2.1 Operation of a Synchronous System

A digital synchronous circuit is a network of functional logic elements and globally clocked registers. A *single-phase* clock signal and *edge-triggered* registers are assumed throughout this paper. For an arbitrary *ordered* pair of registers $\langle R_1, R_2 \rangle$, one of the following two situations can be observed: either (1) the input of R_2 *cannot* be reached from the output of R_1 by propagating through a sequence of logic elements *only;* or (2) there exists at least one sequence of logic elements *only* that connects the output of R_1 to the input of R_2. In the former case—denoted by $R_1 \not\rightsquigarrow R_2$—switching events at the output of R_1 do *not* affect the input of R_2 during the same clock period. In the latter case—denoted by $R_1 \rightsquigarrow R_2$—signal switching at the output of R_1 propagates to the input of R_2. In this case, $\langle R_1, R_2 \rangle$ is called a *sequentially-adjacent* pair of registers which make up a *local data path.*

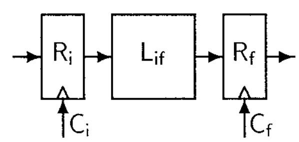


Figure 1: A local data path.

An example of a local data path $R_i \rightsquigarrow R_f$ is shown in Figure 1. The clock signals C_i and C_f synchronize the sequentially-adjacent pair of registers R_i

and R_f, respectively. Signal switching at the output of R_i is triggered by the clock signal C_i and, after propagating through the logic block L_{if}, this data signal appears at the input of R_f and is successfully latched. The minimum and maximum delays through this local data path are called the *short path* and *long path* delays, respectively, and are denoted by $\hat{d}(i,f)$ and $\hat{D}(i,f)$, respectively. Note that both $\hat{d}(i,f)$ and $\hat{D}(i,f)$ are due to the accumulative effects of three sources of delay [1]. These sources are the *clock-to-output* delay of R_i, a delay introduced by the signal propagating through L_{if}, and an interconnect delay due to any wires along the signal path $R_i \rightsquigarrow R_f$.

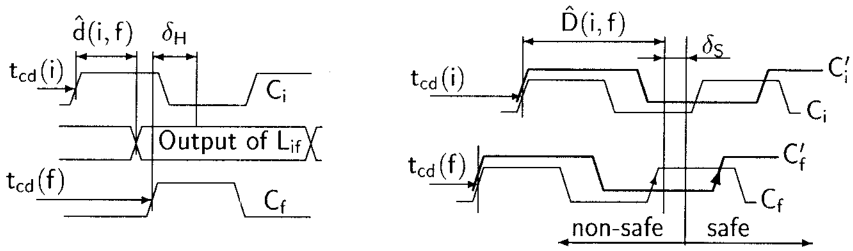
The clock signals C_i and C_f originate at the clock source and are delivered to R_i and R_f by the clock distribution network with delays $t_{cd}(i)$ and $t_{cd}(f)$, respectively (positive edge-triggered registers are assumed). Formally, the difference

$$T_{Skew}(i,f) = t_{cd}(i) - t_{cd}(f) \tag{1}$$

is defined as the *clock skew* $T_{Skew}(i,f)$ between the registers R_i and R_f. In addition, note that $T_{Skew}(i,f)$ as defined in (1) obeys the antisymmetric property,

$$T_{Skew}(i,f) = -T_{Skew}(f,i). \tag{2}$$

The clock skew can be either *negative* or *positive,* as illustrated in Figures 2a and 2b, respectively. Negative clock skew may be used to effectively speed-up a local data path $R_i \rightsquigarrow R_f$ by allowing an extra $T_{Skew}(i,f)$ time for the signal to propagate from R_i to R_f. However, excessive negative skew may create a *clock hazard* or *race condition,* known as *double clocking* [1,15]. Similarly, positive clock skew effectively decreases the clock period T_{CP} by $T_{Skew}(i,f)$, thereby limiting the maximum clock frequency (note clock signals C_i', C_f' in Figure 2b). In this case, a clocking hazard known as *zero clocking* may be created [1,15].



(a) Double clocking due to excessive negative clock skew

(b) Zero clocking due to excessive positive clock skew

Figure 2: Clock skew affecting the performance of a local data path

The clocking hazards illustrated in Figure 2 are avoided if the following timing relationships are satisfied for each local data path $R_i \rightsquigarrow R_f$ [1,15]:

$$t_{cd}(i) + \hat{d}(i,f) > t_{cd}(f) + \delta_H(f) \tag{3}$$

$$t_{cd}(i) + \hat{D}(i,f) + \delta_S(f) < t_{cd}(f) + T_{CP}. \tag{4}$$

The quantities $\delta_H(f)$ and $\delta_S(f)$ denote the register *hold* time and *set-up* time of R_f, respectively. Accounting for (1) and making the following substitutions,

$$\hat{d}(i,f) - \delta_H(f) = d(i,f) \tag{5}$$

$$\hat{D}(i,f) + \delta_S(f) = D(i,f), \tag{6}$$

leads to a simplified form for the constraints (3) and (4):

$$T_{Skew}(i,f) > d(i,f) \tag{7}$$

$$T_{Skew}(i,f) < T_{CP} - D(i,f). \tag{8}$$

Modeling of Synchronous Digital Systems as Graphs Certain properties of a synchronous digital system may be better understood by analyzing a graph model of such a system. A synchronous digital system can be modeled [16, 17] as a *directed graph* G with *vertex set* $V = \{v_1, \dots, v_{N_R}\}$ and *edge set* $E = \{e_1, \dots, e_{N_P}\} \subseteq V \times V$. An example of a circuit graph G is illustrated in Figure 3a. The number of registers in the circuit is $|V| = N_R$ and vertex v_k corresponds to the register R_k. The number of local data paths in the circuit is $|E| = N_P$. An edge is directed from v_i to v_j *iff* $R_i \rightsquigarrow R_j$. In the case where there are multiple paths between a sequentially-adjacent pair of registers $R_i \rightsquigarrow R_j$, only *one* edge connects v_i to v_j. The *underlying* graph G_u of the graph G is a *non-directed* graph that has the same vertex set V, where the directions have been removed from the edges. In Figure 3, an input or an output of the circuit is indicated by an edge incident to only one vertex.

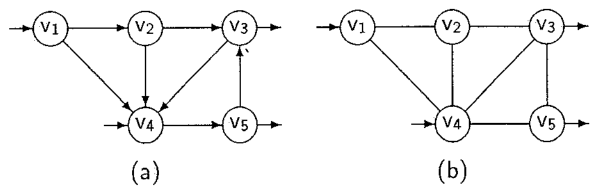


Figure 3: Graph G of a circuit with $N_R = 5$ registers. (a) The *directed* graph G. (b) The *underlying* graph G_u corresponding to the graph G in (a).

2.2 Clock Scheduling

Examining constraints (7) and (8) reveals a procedure for preventing clock hazards. Assuming (8) is not satisfied, a suitably large value of T_{CP} can be chosen to satisfy constraint (8) and prevent zero clocking. This technique is illustrated in Figure 2b—the clock signals C_i' and C_f' have the same delays as C_i and C_f but an increased period. Also note that unlike (8), (7) is independent of T_{CP}. Therefore, T_{CP} cannot be varied to correct a double clocking hazard, but rather a redesign of the clock distribution network may be required [10].

Both double and zero clocking hazards can be eliminated if two simple choices characterizing a digital circuit are made. Specifically, if equal values are chosen for all clock delays and a sufficiently large value—larger than the longest delay—is chosen for T_{CP}, neither clocking hazard will occur. Formally,

$$\forall \langle R_i, R_f \rangle \, : \, t_{cd}(i) = t_{cd}(f) = \text{Const} \quad \text{and} \quad R_i \rightsquigarrow R_f \Rightarrow T_{CP} > D(i,f), \qquad (9)$$

and, with (9), the timing constraints (7) and (8) for a hazard-free local data path $R_i \rightsquigarrow R_f$ become

$$d(i,f) > 0 \qquad (10)$$

$$D(i,f) < T_{CP}. \qquad (11)$$

The choice of the clock period T_{CP} must satisfy (11) and typically, $d(i,f) > 0$ (or equivalently $\hat{d}(i,f) > \delta_H(f)$) for a properly designed local data path, thereby satisfying (10).

The application of (9), (10), and (11) has been central to digital synchronous circuit design methodologies for decades [1, 18]. By requiring the clock delays to each register to be approximately equal, these design methods are known as *zero* clock skew methods. As shown by previous research [1, 8–10, 19–21], both double and zero clocking hazards may be removed from a synchronous digital circuit even when $T_{Skew}(i,f) \neq 0$ for some (or *all*) local data paths $R_i \rightsquigarrow R_f$. As long as (7) and (8) are satisfied, a synchronous digital system can operate reliably with *non-zero* clock skews, permitting the system to operate at higher clock frequencies while removing all race conditions.

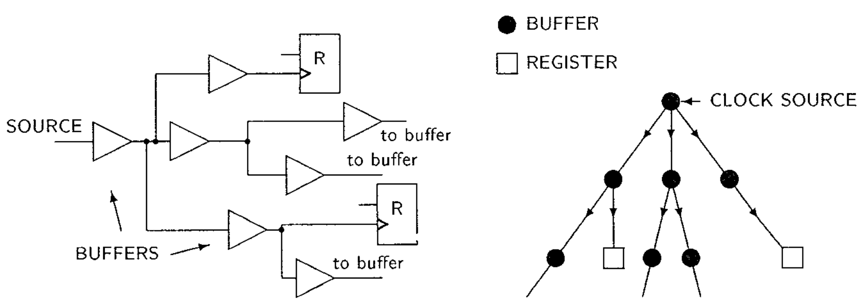
The vector column of clock delays $\mathbf{T_{CD}} = [t_{cd}(1), t_{cd}(2), \ldots]^T$ is called a *clock schedule* [1, 15]. If $\mathbf{T_{CD}}$ is chosen such that (7) and (8) are satisfied for every local data path $R_i \rightsquigarrow R_f$, $\mathbf{T_{CD}}$ is called a *consistent* clock schedule. A clock schedule that satisfies (9) is called a *trivial* clock schedule. Note that a trivial $\mathbf{T_{CD}}$ implies global *zero* clock skew since for any i and f, $t_{cd}(i) = t_{cd}(f)$, thus, $T_{Skew}(i,f) = 0$.

Fishburn first suggested in [15] an algorithm for computing a consistent clock schedule that is *nontrivial*. Furthermore, it was shown in [15] that by exploiting negative and positive clock skew on the local data paths $R_i \rightsquigarrow R_f$, a circuit can operate with a clock period T_{CP} lower than the clock period achievable by a trivial clock schedule that satisfies the conditions represented in (9). In fact, Fishburn [15] determined an *optimal* clock schedule by applying linear programming techniques to solve for $\mathbf{T_{CD}}$ so as to satisfy (7) and (8) while minimizing the objective function $F_{\text{objective}} = T_{CP}$.

The process of determining a consistent clock schedule $\mathbf{T_{CD}}$ can be considered as the mathematical problem of minimizing T_{CP} under the constraints, (7) and (8). However, there are important practical issues to consider before a clock schedule can be properly implemented. A clock distribution network must be synthesized such that the clock signal is delivered to each register with the proper delay so as to satisfy the clock skew schedule $\mathbf{T_{CD}}$. Furthermore, this clock distribution network must be constructed so as to minimize the deleterious effects of *interconnect impedances* and *process parameter variations* on the implemented clock schedule. Synthesizing the clock distribution network typically consists of determining a *topology* for the network, together with the circuit design and physical layout of the *buffers* and *interconnect* within the clock distribution network [1].

2.3 Structure of the Clock Distribution Network

The clock distribution network is typically organized as a rooted tree structure [1, 8, 16], as illustrated in Figure 4, and is often called a *clock tree* [1]. A circuit schematic of a clock distribution network is shown in Figure 4a. An abstract graphical representation of the tree structure in Figure 4a is shown in Figure 4b. The unique source of the clock signal is at the root of the



(a) Circuit structure of the clock distribution network

(b) Clock tree structure that corresponds to the circuit shown in (a)

Figure 4: Tree structure of a clock distribution network

tree. This signal is distributed from the source to every register in the circuit through a sequence of buffers and interconnect. Typically, a buffer in the network drives a combination of other buffers and registers in the pipelined DSP circuit. An interconnection network of wires connects the output of the driving buffer to the inputs of these driven buffers and registers. A buffer is typically placed in each branch of the clock tree and the registers are the *leaves* of the tree. There are N_R leaves in the clock tree labeled F_1 through F_{N_R} where leaf F_j corresponds to register R_j.

A clock tree topology that implements a given clock schedule $\mathbf{T_{CD}}$ must enforce a clock skew $T_{Skew}(i,f)$ for each local data path $R_i \rightsquigarrow R_f$ of the circuit in order to ensure that both (7) and (8) are satisfied. This topology, however, can be profoundly affected by three important issues relating to the operation of a fully synchronous digital system.

Issue 1: Previous research [10, 21] has indicated that tight control over the clock skews rather than the clock delays is necessary for the circuit to operate reliably. Equations (7) and (8) are used in [21] to determine a *permissible range* of the allowed clock skew for each local data path. The concept of a permissible range for the clock skew $T_{Skew}(i,f)$ of a local data path $R_i \rightsquigarrow R_f$ is illustrated in Figure 5. When $T_{Skew}(i,f) \in [d(i,f), T_{CP} - D(i,f)]$—as shown in Figure 5—(7) and (8) are satisfied. $T_{Skew}(i,f)$ is not *permitted* to be in either the interval $(-\infty, -d(i,f))$ because a race condition will be created or the interval $(T_{CP} - D(i,f), +\infty)$ because the minimum clock period will be limited.

Issue 2: An important corollary related to the *conservation property* [1] of clock skew is that there is a *linear dependency* among the clock skews of a global data path that form a cycle in the underlying graph of the circuit. Specifically, if $v_0, e_1, v_1 (\neq v_0), \ldots, v_{k-1}, e_k, v_k \equiv v_0$ is a cycle in the

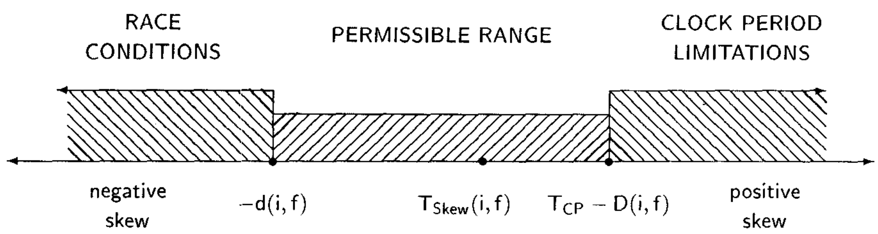


Figure 5: The permissible range of the clock skew of a local data path $R_i \rightsquigarrow R_f$. A clock hazard exists if $T_{Skew}(i,f) \notin [d(i,f), T_{CP} - D(i,f)]$.

underlying graph of the circuit, then

$$0 = [t_{cd}(0) - t_{cd}(1)] + [t_{cd}(1) - t_{cd}(2)] + \ldots + [t_{cd}(k-1) - t_{cd}(0)]$$
$$= \sum_{i=0}^{k-1} T_{Skew}(i, i+1). \qquad (12)$$

The importance of this property is that (12) describes the inherent correlation among certain clock skews within a circuit. Therefore, these correlated clock skews *cannot* be optimized independently of each other. Returning to Figure 3, note that it is not necessary that a cycle exists in the directed graph of a circuit for (12) to hold. For example, v_2, v_3, v_4 is not a cycle in the graph G in Figure 3a but v_2, v_3, v_4 *is* a cycle in the graph G_u in Figure 3b. In addition, $T_{Skew}(2,3) + T_{Skew}(3,4) + T_{Skew}(4,2) = 0$, *i.e.*, the skews $T_{Skew}(2,3)$, $T_{Skew}(3,4)$, and $T_{Skew}(4,2)$ are linearly dependent. A maximum of $|V| - 1 = N_R - 1$ clock skews can be chosen independently of each other in a circuit, which is easily proven by considering a spanning tree of the underlying circuit graph G_u. Any spanning tree of G_u will contain $N_R - 1$ edges—each edge corresponding to a local data path—and the addition of any other edge of G_u will form a cycle for which (12) holds. Note, for example, that for the circuit modeled by the graph in Figure 3, *four* independent clock skews can be chosen such that the remaining three clock skews can be expressed in terms of the independent clock skews.

Issue 3: The clock signal delay $t_{cd}(j)$ from the clock source to R_j is equal to the sum of the propagation delays of the buffers on the unique path that exists between the root and F_j. Furthermore, if $R_i \rightsquigarrow R_f$ is a sequentially-adjacent pair of registers, there is a portion of the two paths—denoted P^*_{if}—between the root of the clock tree and R_i and R_f, respectively, that is common to both paths. Similarly, there is a portion of the path to any of the registers R_i and R_f in a sequentially-adjacent pair, denoted by P^i_{if} and P^f_{if}, respectively, that is unique to this register. Therefore, the clock skew $T_{Skew}(i,f)$ between R_i and R_f is equal to the difference between the accumulated buffer propagation delays between P^i_{if} and P^f_{if}, *i.e.*, $T_{Skew}(i,f) = \text{Delay}(P^i_{if}) - \text{Delay}(P^f_{if})$. Therefore, any variations of circuit parameters over P^*_{if} will not affect $T_{Skew}(i,f)$.

This differential feature of the clock tree suggests an approach for minimizing the effects of process parameter variations on the correct operation of the circuit. This approach is based on choosing a structure for the clock tree that restricts the possible variations of those local data paths with narrow permissible ranges, and tolerates larger delay variations for those local data paths with wider permissible ranges. It is advantageous to maximize P^*_{if} for any local data path $R_i \rightsquigarrow R_f$ with a narrow permissible range, such that the parameter variations on P^*_{if} would have no effect on $T_{Skew}(i, f)$. Similarly, when the permissible range $[-d(i, f), T_{CP} - D(i, f)]$ is wider, P^*_{if} may be permitted to be only a small fraction of the total path from the root to R_i and R_f, respectively.

3 SOLUTION AND EXPERIMENTAL RESULTS

A solution to the topological synthesis problem is presented in this section. This solution is based on the following assumption: *the signal propagation delay through a node is a constant, denoted by* Δ_b. This constant Δ_b delay can be achieved if, for example, the load of every buffer in the clock tree is kept equal throughout the tree. In such a case, the propagation delay δ_j of the clock signal from the clock source to the register R_j at depth b_j is $t_{cd}(j) = \delta_j = b_j \times \Delta_b$. The load of a buffer in the clock tree can be kept constant if the total fanout and interconnect capacitance is kept constant.

After substituting this expression into (7) and (8), the necessary conditions to avoid either clock hazard can be rewritten as follows:

$$T_{Skew}(i, f) = (b_i - b_f)\Delta_b > -d(i, f) \tag{13}$$

$$-T_{Skew}(i, f) = (b_f - b_i)\Delta_b > D(i, f) - T_{CP}. \tag{14}$$

Therefore, the problem of designing the topology of the clock distribution network can be formulated as an optimization problem of minimizing T_{CP} subject to the constraints (13) and (14).

The quantities b_i and b_f are integers, since these terms denote the number of branches (buffers) from the root of the clock tree to a particular leaf (*i.e.*, register). In the general case, this optimization problem can be described as a *mixed-integer* linear programming problem (since T_{CP} can be any real positive number), and is difficult to solve. However, previous research has demonstrated [22] that if a fixed value for the clock period T_{CP} is chosen, the problem changes as follows. Given a value for T_{CP}, find a set of integers $\{b_1, b_2, \ldots, b_i, \ldots\}$ such that

$$\begin{aligned} (b_i - b_j)\Delta_b &> -d(i, j) \\ \text{and } (b_j - b_i)\Delta_b &> D(i, j) - T_{CP} \end{aligned} \tag{15}$$

for every sequentially-adjacent pair of registers $R_i \rightsquigarrow R_j$, or determine that no such set of integers exist. Once (15) has been solved for a particular circuit, a clock tree topology such as the network shown in Figure 4 can be implemented.

Each register R_i of the circuit receives its clock signal from a leaf F_i of the clock tree at a branching depth $b = b_i$, where b_i is the integer obtained from solving (15). In addition, Leiserson and Saxe describe in [23] an algorithm for efficiently solving similar optimization problems such as represented by (15). The run time of this algorithm is $O(V(E + V) \log V)$, where V and E denote the number of registers and the number of sequentially-adjacent pairs of registers, respectively. This algorithm is applied in this synthesis methodology for constructing the topology of the clock tree.

After computing the clock schedule, a mapping $M : t_{cd} \mapsto B$ is produced such that each clock delay $t_{cd}(i)$ is mapped to a non-negative *integer* number $b(i) \in B = \{1, 2, \dots, b_{max}\}$. The integer $b(i)$ is the required depth of the leaf in the clock tree driving the register R_i. Typically, $b_{max} < N_R$, since there may be more than one register with the same value of the required depth b. In addition, note that the set B can be redefined as $\{1 + k, 2 + k, \dots, b_{max} + k\}$ without affecting the validity of the solution (k is any integer). For example, if the solution for a circuit with 10 registers is $b(1), \dots, b(10) = \{3, 5, 8, 10, -2, 0, 0, 5, 5, 4\}$, this solution can be changed to $\{5, 7, 10, 12, 0, 2, 2, 7, 7, 6\}$ by adding two branches (or buffers) to each of the numbers $b(1)$ through $b(10)$.

The clock distribution network is implemented recursively in the following manner. An integer value called the *branching factor* f is initially chosen. The branching factor determines the number of outgoing branches from each node of the clock tree. By maintaining f constant throughout the clock tree, the requirement for constant Δ_b can be satisfied. A specific number of registers n_j is driven at a specific depth $b(j)$ of the clock tree. Therefore, at least $\lceil n_j / f \rceil$ buffers at depth $b(j-1)$ of the clock tree are required to drive these n_j registers at depth $b(j)$. The number of required buffers and branches in the clock tree is determined by beginning at the bottom of the tree (those leaves with the greatest depth) and recursively computing the number of buffers at each preceding level.

Table 1: ISCAS'89 suite of circuits. The name, number of registers, bounds of the searchable clock period, optimal clock period (T_{opt}), and performance improvement (in per cent) are shown for each circuit. Also shown in the last two columns labeled B_2 and B_3, respectively, are the number of buffers in the clock tree for $f = 2$ and $f = 3$, respectively.

Circuit	Regs	T_{min}	T_{max}	T_{opt}	% Imp.	B_2	B_3	Circuit	Regs	T_{min}	T_{max}	T_{opt}	% Imp.	B_2	B_3
s1196	18	7.80	20.80	13.00	17%	21	14	s400	21	8.40	14.20	8.88	37%	25	14
s13207	669	60.40	85.60	60.45	29%	681	348	s420.1	16	5.20	16.40	7.45	55%	21	15
s1423	74	75.80	92.20	79.00	14%	80	45	s444	21	8.40	16.80	10.17	39%	23	15
s1488	6	31.00	32.20	31.00	4%	5	4	s510	6	14.80	16.80	15.20	10%	7	5
s15850	597	83.60	116.00	83.98	28%	614	320	s526	21	9.40	13.00	10.48	19%	21	10
s208.1	8	5.20	12.40	5.48	56%	10	9	s526n	21	9.40	13.00	10.48	19%	21	10
s27	3	5.40	6.60	5.40	18%	3	3	s5378	179	20.40	28.40	22.29	22%	182	93
s298	14	9.40	13.00	10.48	19%	13	8	s641	19	71.00	88.00	71.03	19%	30	22
s344	15	18.40	27.00	18.65	31%	16	11	s713	19	79.20	89.20	72.23	19%	31	23
s349	15	18.40	27.00	18.65	31%	15	10	s820	5	19.20	19.20	19.20	0%	11	9
s35932	1728	34.20	34.20	34.20	0%	3457	2595	s832	5	19.80	19.80	19.80	0%	11	9
s382	21	8.00	14.20	8.88	37%	25	14	s838.1	32	5.20	24.40	8.76	64%	40	24
s38417	1636	42.20	69.00	42.82	38%	1647	832	s9234.1	211	54.20	75.80	54.24	28%	220	113
s38584	1452	67.60	94.20	67.65	28%	1465	743	s9234	228	54.20	75.80	54.24	28%	237	123
s386	6	17.00	17.80	17.80	0%	12	10	s953	29	16.40	23.20	18.96	18%	31	18

The algorithm has been implemented in a 3,300 line program written in the C++ high-level programming language. This program has been executed on the ISCAS'89 suite of benchmark circuits. A summary of the results for these benchmark circuits is shown in Table 1. These results demonstrate that by applying the proposed algorithm to schedule the clock delays to each register, up to a 64% decrease in the minimum clock period can be achieved for these benchmark circuits while removing any race conditions.

For example, an implementation of the clock tree topology of the circuit s1423 is shown in Figure 6. The branching factor for this clock tree is $f = 3$.

The circuit s1423 contains 74 registers. A 14% improvement in the minimum clock period is demonstrated on this circuit by applying the methodology described in this paper.

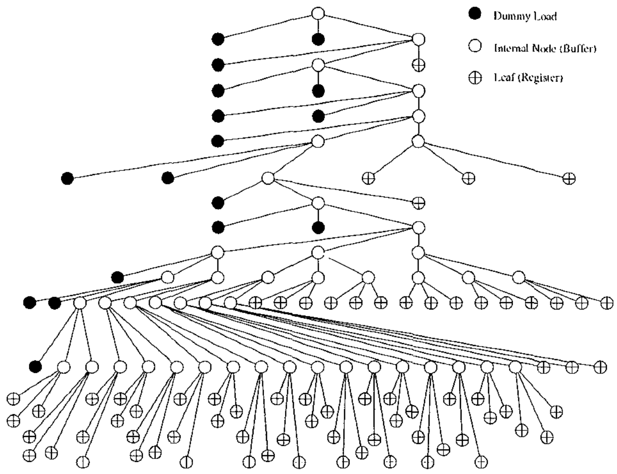


Figure 6: Buffered clock tree for the benchmark circuit s1423. The circuit s1423 has a total of 74 registers and the clock tree consists of 45 buffers when the branching factor is $f = 3$.

4 CONCLUSIONS

The problem of synthesizing the topology of a buffered clock distribution network from a clock skew schedule is examined in this paper. A new, integer linear programming approach, based on local timing information, is presented for simultaneously determining an acceptable non-zero clock skew schedule and a topology of the clock distribution network that minimizes the clock period and efficiently builds the clock tree.

5 ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under Grant No. MIP-9208165 and Grant No. MIP-9423886, the Army Research Office under Grant No. DAAH04-93-G-0323, a grant from the New York State Science and Technology Foundation to the Center for Advanced Technology-Electronic Imaging Systems, and by grants from the Xerox Corporation, IBM Corporation, and Intel Corporation.

References

[1] E. G. Friedman, *Clock Distribution Networks in VLSI Circuits and Systems.* IEEE Press, 1995.

[2] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley Publishing Company, 1990.

[3] S. Bothra, B. Rogers, M. Kellam, and C. M. Osburn, "Analysis of the Effects of Scaling on Interconnect Delay in ULSI Circuits," *IEEE Transactions on Electron Devices*, Vol. ED-40, No. 3, pp. 591–597, March 1993.

[4] W. J. Bowhill *et al.*, "Circuit Implementation of a 300-MHz 64-bit Second-generation CMOS Alpha CPU," *Digital Technical Journal*, Vol. 7, No. 1, pp. 100–118, 1995.

[5] N. Gaddis and J. Lotz, "A 64-b Quaf-Issue CMOS RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, Vol. SC-31, No. 11, pp. 1697–1702, November 1996.

[6] P. E. Gronowski *et al.*, "A 433-MHz 64-bit Quad-Issue RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, Vol. SC-31, No. 11, pp. 100–118, November 1996.

[7] N. Vasseghi, K. Yeager, E. Sarto, and M. Seddighnezhad, "200-Mhz Superscalar RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, Vol. SC-31, No. 11, pp. 1675–1686, November 1996.

[8] J. L. Neves and E. G. Friedman, "Topological Design of Clock Distribution Networks Based on Non-Zero Clock Skew Specification," *Proceedings of the 36th IEEE Midwest Symposium on Circuits and Systems*, pp. 468–471, August 1993.

[9] J. G. Xi and W. W.-M. Dai, "Useful-Skew Clock Routing With Gate Sizing for Low Power Design," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 383–388, June 1996.

[10] J. L. Neves and E. G. Friedman, "Design Methodology for Synthesizing Clock Distribution Networks Exploiting Non–Zero Localized Clock Skew," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. VLSI-4, No. 2, pp. 286–291, June 1996.

[11] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "Clock Routing for High-Performance ICs," *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 573–579, June 1990.

[12] R.-S. Tsay, "An Exact Zero-Skew Clock Routing Algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-12, No. 2, pp. 242–249, February 1993.

[13] N.-C. Chou and C.-K. Cheng, "On General Zero-Skew Clock Net Construction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. VLSI-3, No. 1, pp. 141–146, March 1995.

[14] N. Ito, H. Sugiyama, and T. Konno, "ChipPRISM: Clock Routing and Timing Analysis for High-Performance CMOS VLSI Chips," *Fujitsu Scientific and Technical Journal*, Vol. 31, pp. 180–187, December 1995.

[15] J. P. Fishburn, "Clock Skew Optimization," *IEEE Transactions on Computers*, Vol. C–39, No. 7, pp. 945–951, July 1990.

[16] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* MIT Press, 1989.

[17] D. B. West, *Introduction to Graph Theory.* Prentice-Hall Inc., 1996.

[18] T.-C. Lee and J. Kong, "The new line in IC Design," *IEEE Spectrum*, pp. 52–58, March 1997.

[19] E. G. Friedman, "The Application of Localized Clock Distribution Design to Improving the Performance of Retimed Sequential Circuits," *Proceedings of the IEEE Asia–Pacific Conference on Circuits and Systems*, pp. 12–17, December 1992.

[20] I. S. Kourtev and E. G. Friedman, "Simultaneous Clock Scheduling and Buffered Clock Tree Synthesis," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1812–1815, June 1997.

[21] J. L. Neves and E. G. Friedman, "Optimal Clock Skew Scheduling Tolerant to Process Variations," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 623–628, June 1996.

[22] R. B. Deokar and S. S. Sapatnekar, "A Graph–Theoretic Approach to Clock Skew Optimization," *Proceedings of the IEEE International Symposium on Circuits and Systems*, Vol. 1, pp. 407–410, May 1994.

[23] C. E. Leiserson and J. B. Saxe, "A Mixed-Integer Linear Programming Problem Which is Efficiently Solvable," *Journal of Algorithms*, Vol. 9, pp. 114–128, March 1988.