Synthesis of clock tree topologies to implement nonzero clock skew schedule

I.S.Kourtev and E.G.Friedman

Abstract: Designing the topology of a clock distribution network is considered for a synchronous digital integrated circuit so as to satisfy a nonzero clock skew schedule. A methodology and related algorithms for synthesising 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 problem. The approach is demonstrated on the suite of ISCAS'89 benchmark circuits. Up to 64% performance improvement is attained on these circuits by exploiting nonzero clock skew throughout the synchronous system. Clock tree topologies that implement the nonzero clock skew schedule based on the synthesis algorithms presented are described for each of the benchmark circuits.

1 Introduction

Most high performance digital integrated circuits implement data processing algorithms based on the iterative execution of simple operations. Typically, these algorithms are highly parallelised and pipelined by inserting clocked registers at specific locations throughout the circuit. The synchronisation strategy for these clocked registers in the vast majority of VLSI/ULSI-based digital systems is a fully synchronous approach. It is not uncommon for the computational process in these systems to be spread over hundreds of thousands of functional logic elements and tens of thousands of registers.

For such 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 synchronisation signal, known as the clock signal. For a fully synchronous system to operate correctly, 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 a 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 systems. The primary source of the load for the clock signals has shifted 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 increase 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 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 synthesising the topology of a nonzero skew clock distribution network that satisfies the tighter timing constraints required in high performance VLSI-complexity systems.

2 Background

In this Section 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 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 analysed.

2.1 Operation of 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 the input of R_2 cannot be reached from the output of R_1 by propagating through a sequence of logic elements only; or there exists at least one sequence of logic elements only that connects the output of R_1 to the

IEE Proceedings online no. 19990582

DOI: 10.1049/ip-cds:19990582

Paper first received 29th May 1998 and in revised form 4th May 1999

I.S. Kourtev was with the University of Rochester and is now with the Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA

E.G. Friedman is with the Department of Electrical and Computer Engineering, University of Rochester, P.O. Box 270231, Rochester, NY 14627-0231, USA

input of R_2. In the former case, denoted by $R_1 \not\rightarrow R_2$, 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 $R_1 \rightarrow R_2$, 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.

An example of a local data path $R_i \rightarrow R_f$ with flip-flops is shown in Fig. 1. The clock signals C_i and C_f synchronise 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 $R_i \rightarrow R_f$.

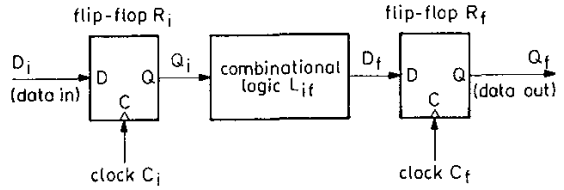

Fig. 1 *Local data path with flip-flops*

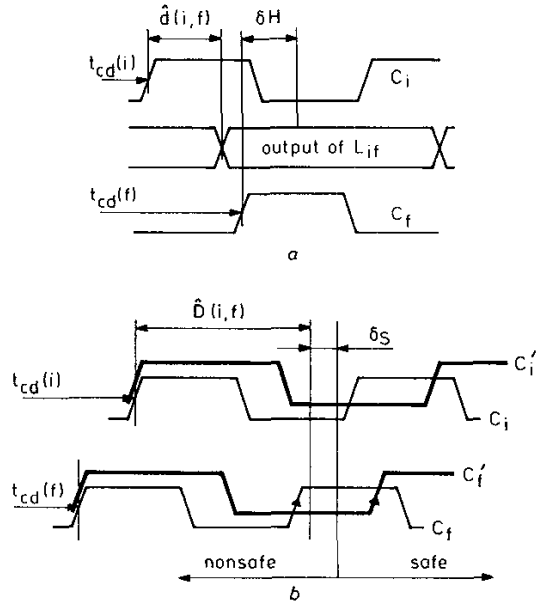

Fig. 2 *Clocking hazards of local data path*
a Double clocking due to excessive negative clock skew
b Zero clocking due to excessive positive clock skew

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

$$T_{skew}(i, f) = t_{cd}(i) - t_{cd}(f) \quad (1)$$

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 eqn. 1 obeys the antisymmetric property

$$T_{skew}(i, f) = -T_{skew}(f, i) \quad (2)$$

The clock skew can be either negative or positive, as illustrated in Figs. 2*a* and *b*, respectively. Negative clock skew may be used to effectively speed-up a local data path $R_i \rightarrow R_f$ 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 a 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 Fig. 2*b*). In this case, a clocking hazard known as zero clocking may be created [1, 15].

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

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

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

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

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

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

yields a simplified form for the constraints of eqns. 3 and 4.

$$T_{skew}(i, f) > -d(i, f) \quad (7)$$

$$T_{skew}(i, f) < T_{CP} - D(i, f) \quad (8)$$

2.1.1 Modelling of synchronous digital systems as graphs.:

Certain properties of a synchronous digital system may be better understood by analysing a graph model of such a system. A synchronous digital system can be modelled [16, 17] as a directed graph G with vertex set $V = \{v_1, ..., v_{N_R}\}$ and edge set $E = \{e_1, ..., e_{N_R}\} \subseteq V \times V$. An example of a circuit graph G is illustrated in Fig. 3*a*. 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 $R_i \rightarrow R_j$. In the case where there are multiple paths between a sequentially-adjacent pair of registers $R_i \rightarrow R_j$ only one edge connects v_i to v_j. The underlying graph G_u of the graph G is a nondirected graph that has the same vertex set V, where the directions have been removed from the edges. In Fig. 3 an input or an output of the circuit is indicated by an edge incident to only one vertex.

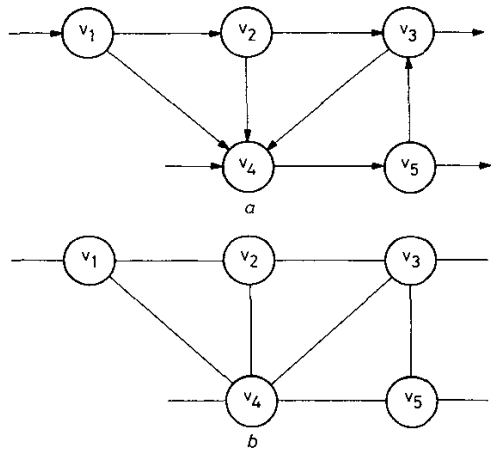

Fig. 3 *Graph G of circuit with $N_R = 5$ registers*
a Directed graph G
b Underlying graph G_u corresponding to graph G in *a*

2.2 Clock scheduling

Examining the constraints of eqns. 7 and 8 reveals a procedure for preventing clock hazards. Assuming eqn. 8 is not satisfied, a suitably large value of T_{CP} can be chosen to satisfy eqn. 8 and prevent zero clocking. This technique is illustrated in Fig. 2*b*; 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 eqn. 8, eqn. 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 characterising 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

$$\forall\langle R_i, R_f\rangle : t_{cd}(i) = t_{cd}(f) = \text{const. and } R_i \rightarrow R_f \Rightarrow D(i,f) < T_{CP} \quad (9)$$

and, with eqn. 9, the timing constraints of eqns. 7 and 8 for a hazard-free local data path $R_i \rightarrow R_f$ become

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

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

The choice of the clock period T_{CP} must satisfy eqn. 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 eqn. 10.

The application of eqns. 9–11 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 $R_i \rightarrow R_f$. As long as eqns. 7 and 8 are satisfied, a synchronous digital system can operate reliably with nonzero clock skews, permitting the system to operate at higher clock frequencies while removing all race conditions.

The vector column of clock delays $\boldsymbol{T_{CD}} = [t_{cd}(1), t_{cd}(2), \ldots]^T$ is called a clock schedule [1, 15]. If $\boldsymbol{T_{CD}}$ is chosen such that eqns. 7 and 8 are satisfied for every local data path $R_i \rightarrow R_f$, $\boldsymbol{T_{CD}}$ is called a consistent clock schedule. A clock schedule that satisfies eqn. 9 is called a trivial clock schedule. Note that a trivial $\boldsymbol{T_{CD}}$ 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 [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 $R_i \rightarrow R_f$, a circuit can operate with a clock period T_{CP} less than the clock period achievable by a trivial clock schedule that satisfies the conditions represented in eqn. 9. In fact, Fishburn [15] determined an optimal clock schedule by applying linear programming techniques to solve for $\boldsymbol{T_{CD}}$ so as to satisfy eqns. 7 and 8 while minimising the objective function $F_{objective} = T_{CP}$.

The process of determining a consistent clock schedule $\boldsymbol{T_{CD}}$ can be considered as the mathematical problem of minimising T_{CP} under the constraints of eqns. 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 synthesised such that the clock signal is delivered to each register with the proper delay so as to satisfy the clock skew schedule $\boldsymbol{T_{CD}}$. Furthermore, this clock distribution network must be constructed so as to minimise the deleterious effects of interconnect impedances and process parameter variations on the implemented clock schedule. Synthesising 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 clock distribution network

The clock distribution network is typically organised as a rooted tree structure [1, 8, 16], as illustrated in Fig. 4, and is often called a clock tree [1]. A circuit schematic of a clock distribution network is shown in Fig. 4*a*. An abstract graphical representation of the tree structure in Fig. 4*a* is shown in Fig. 4*b*. The unique source of the clock signal is at the root of the 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 circuit. An interconnection network of wires connects the output of the driving buffer to the inputs of these driven buffers and registers. An internal node of the tree corresponds to a buffer and a leaf node of the tree corresponds to a register. There are N_R leaves in the clock tree labelled F_1 through F_{N_R} where leaf F_j corresponds to register R_j.

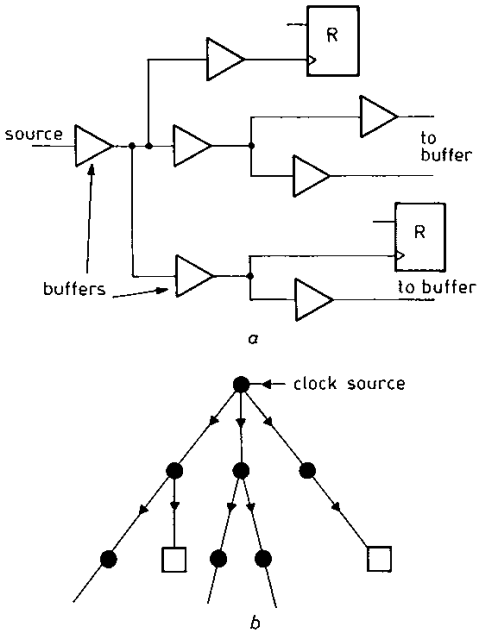

Fig. 4 *Tree structure of clock distribution network*
a Circuit structure of clock distribution network
b Clock tree structure corresponding to circuit shown in *a*
● buffer
□ register

A clock tree topology that implements a given clock schedule $\boldsymbol{T_{CD}}$ must enforce a clock skew $T_{skew}(i,f)$ for each local data path $R_i \rightarrow R_f$ of the circuit to ensure that both eqns. 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.

2.3.1 Issue 1: 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 $v_0, e_1, v_1 (\neq v_0), ..., v_{k-1}, e_k, v_k \equiv v_0$ is a cycle in the underlying graph of the circuit, then

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

The importance of this property is that eqn. 12 describes the inherent correlation among certain clock skews within a circuit. Therefore these correlated clock skews cannot be optimised independently of each other. Returning to Fig. 3, note that it is not necessary that a cycle exists in the directed graph of a circuit for eqn. 12 to hold. For example, v_2, v_3, v_4 is not a cycle in the graph G in Fig. 3*a* but v_2, v_3, v_4 is a cycle in the graph G_u in Fig. 3*b*. 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 eqn. 12 holds. Note, for example, that for the circuit modelled by the graph in Fig. 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.

2.3.2 Issue 2: 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. Eqns. 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 $R_i \rightarrow R_f$ is illustrated in Fig. 5. When $T_{skew}(i, f) \in [d(i, f), T_{CP} - D(i, f)]$ as shown in Fig. 5 eqns. 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.

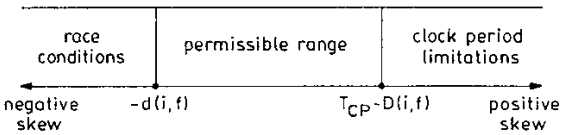

Fig. 5 *Permissible range of clock skew of local data path $R_i \rightarrow R_f$*
$T_{skew}(i, f) = t_{cd}(i) - t_{cd}(f)$
A clock hazard exists if $T_{skew}(i, f) \notin [d(i, f), T_{CP} - D(i, f)]$

2.3.3 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 $R_i \rightarrow R_f$ 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_{if}^i and P_{if}^f, 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_{if}^i and P_{if}^f i.e. $T_{skew}(i, f) = \text{delay}(P_{if}^i) - \text{delay}(P_{if}^f)$. 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 minimising 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 maximise P_{if}^* for any local data path $R_i \rightarrow R_f$ 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

3.1 Description of algorithm

The algorithm presented in this Section is based on the following assumption: the signal propagation delay through a node and all of its children nodes is a constant, denoted by Δ_b. Therefore the propagation delay δ_j of the clock signal from the clock source to the register R_j at depth b_j is $t_{cd}(j) = \delta_j = b_j\Delta_b$. Note that Δ_b includes the delay through both a buffer and the interconnect branches connected to the buffer output. There can be considerable difficulty in practically achieving a constant Δ_b throughout all levels of the clock tree. Therefore current research is focused on removing this constraint by providing variable branch delays.

After substituting $\delta_j = b_j\Delta_b$ into eqns. 7 and 8, the necessary conditions to avoid either clock hazard can be rewritten as follows:

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

$$-T_{skew}(i, f) = (b_f - b_i)\Delta_b > D(i, f) - T_{CP} \quad (14)$$

Therefore the problem of designing the topology of the clock distribution network can be formulated as the optimisation problem of minimising T_{CP} subject to the constraints eqns. 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 optimisation 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 $\{b_1, b_2, ..., b_i, ...\}$ such that

$$(b_i - b_j)\Delta_b > -d(i, j) \quad \text{and} \quad (b_j - b_i)\Delta_b > D(i, j) - T_{CP} \quad (15)$$

for every sequentially-adjacent pair of registers $R_i \rightarrow R_f$ or determine that no such set of integers exist. Once eqn. 15 has been solved for a particular circuit, a clock tree topology such as the network shown in Fig. 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 eqn. 15. In addition, Leiserson and Saxe describe in [23] an algorithm for efficiently solving similar optimisation problems such as represented by eqn. 15. The run time of this algorithm is $O(VE)$, where V and E denote the number of registers and the number of sequentially-adjacent pairs of registers,

respectively. This algorithm is the algorithm applied in this synthesis methodology for constructing the topology of the clock tree.

The sequence of operations is as follows. A feasible range for the clock period $[T_{min}, T_{max}]$ to be searched is determined initially; the bounds T_{min} and T_{max} are determined as described in [21]. A binary search for the optimal clock period T_{opt} is then performed over the feasible range of the clock period.

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 nonnegative integer number $b(i) \in B = \{1, 2, ..., b_{max}\}$. 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 $\{1 + k, 2 + k, ..., b_{max} + k\}$ without affecting the validity of the solution (k is any integer). For example, if the solution for a circuit with ten registers is $b(1), ..., b(10) = \{3, 5, 8, 10, -2, 0, 0, 5, 5, 4\}$, 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 $\lceil n/f \rceil$ 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.

3.2 ISCAS'89 benchmark circuits results

The algorithm described has been implemented in a 3,300 line program written in the C++ high-level programming language. This program has been executed on a Sun UltraSparc 1 workstation (170MHz) for the ISCAS'89 suite of benchmark circuits. A simple delay model based on the load of a gate is used to extrapolate the gate delays since these benchmark circuits do not contain delay information. A summary of the results for the 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. Due to the relatively large number of buffers required in the clock tree this approach is only practical for circuits with a large number of registers. Achieving zero clock skew in smaller circuits is usually not a significant problem.

One example implementation of the clock tree topology of the circuit is shown in Fig. 6 for the circuit s400. The branching factor for this clock tree example is $f = 3$. The circuit s400 contains 21 registers. As shown in Table 1, the minimum clock period of the circuit s400 can be improved by 37%.

A second example implementation of the clock tree topology of the circuit s1423 is shown in Fig. 7. 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.

Table 1: ISCAS'89 Suite of circuits. Name number of registers, bounds of searchable clock period, optimal clock period (T_{opt}), and performance improvement are shown for each circuit

Circuit	Run time	Regs	T_{min}	T_{max}	T_{opt}	% Imp.	B_2	B_3
s1196	0.03	18	7.80	20.80	13.00	17%	21	14
s13207	15.89	669	60.40	85.60	60.45	29%	681	348
s1423	1.14	74	75.80	92.20	79.00	14%	80	45
s1488	0.05	6	31.00	32.20	31.00	4%	5	4
s15850	58.84	597	83.60	116.00	83.98	28%	614	320
s208.1	0.02	8	5.20	12.40	5.48	56%	10	9
s27	0.04	3	5.40	6.60	5.40	18%	3	3
s298	0.04	14	9.40	13.00	10.48	19%	13	8
s344	0.05	15	18.40	27.00	18.65	31%	16	11
s349	0.04	15	18.40	27.00	18.65	31%	15	10
s35932	—	1728	34.20	34.20	34.20	0%	3457	2595
s382	0.06	21	8.00	14.20	8.88	37%	25	14
s38417	407.81	1636	42.20	69.00	42.82	38%	1647	832
s38584	196.17	1452	67.60	94.20	67.65	28%	1465	743
s386	0.01	6	17.00	17.80	17.80	0%	12	10
s400	0.07	21	8.40	14.20	8.88	37%	25	14
s420.1	0.04	16	5.20	16.40	7.45	55%	21	15
s444	0.06	21	8.40	16.80	10.17	39%	23	15
s510	0.02	6	14.80	16.80	15.20	10%	7	5
s526	0.06	21	9.40	13.00	10.48	19%	21	10
s526n	0.08	21	9.40	13.00	10.48	19%	21	10
s5378	1.84	179	20.40	28.40	22.29	22%	182	93
s641	0.06	19	71.00	88.00	71.03	19%	30	22
s713	0.05	19	79.20	89.20	72.23	19%	31	23
s820	—	5	19.20	19.20	19.20	0%	11	9
s832	—	5	19.80	19.80	19.80	0%	11	9
s838.1	0.25	32	5.20	24.40	8.76	64%	40	24
s9234.1	4.41	211	54.20	75.80	54.24	28%	220	113
s9234	4.96	228	54.20	75.80	54.24	28%	237	123
s953	0.07	29	16.40	23.20	18.96	18%	31	18

Last two columns, labelled B_2 and B_3, respectively, are number of buffers in clock tree for $f = 2$ and $f = 3$, respectively. Shown in second column from left is run time for execution of program

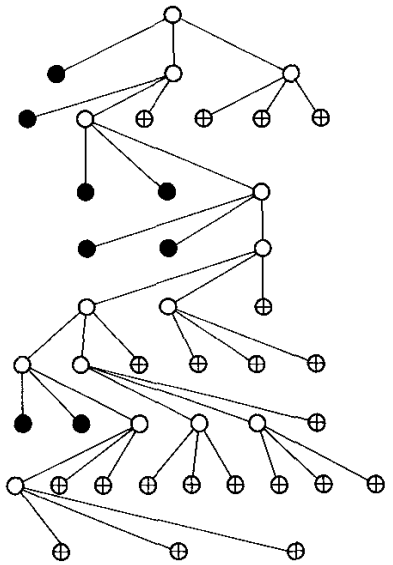

Fig. 6 *Buffered clock tree for benchmark circuit s400*
Circuit s400 has a total of 21 registers and clock tree consists of 14 buffers when branching factor is $f = 3$
● dummy load
○ internal node (buffer)
⊕ leaf (register)

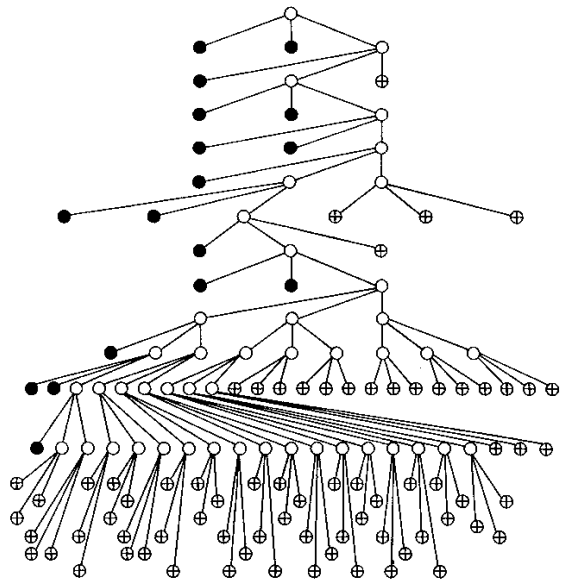

Fig. 7 *Buffered clock tree for benchmark circuit s1423*
Circuit s1423 has total of 74 registers and clock tree consists of 45 buffers when branching factor is $f = 3$
● dummy load
○ internal node (buffer)
⊕ leaf (register)

4 Conclusions

The problem of synthesising the topology of a buffered clock distribution network from a clock skew schedule has been examined. A new, integer linear programming approach based on local timing information has been presented for simultaneously determining an acceptable nonzero clock skew schedule and a topology of the clock distribution network that minimises the clock period and efficiently builds the clock tree.

5 Acknowledgments

This research was supported in part by the National Science Foundation under grants MIP-9208165, MIP-9423886 and MIP-9610108, the Army Research Office under grant DAAH04-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.

6 References

1 FRIEDMAN, E.G.: 'Clock distribution networks in VLSI circuits and systems' (IEEE Press, 1995)
2 BAKOGLU, H.B.: 'Circuits, interconnections, and packaging for VLSI' (Addison-Wesley, Reading, MA, 1990)
3 BOTHRA, S., ROGERS, B., KELLAM, M., and OSBURN, C.M.: 'Analysis of the effects of scaling on interconnect delay in ULSI circuits', *IEEE Trans. Electron Devices*, 1993, **40**, pp. 591–597
4 BOWHILL, W.J.: 'Circuit implementation of a 300-MHz 64-bit second-generation CMOS alpha CPU', *Digital Tech. J.*, 1995, **7**, (1), pp. 100–118
5 GADDIS, N., and LOTZ, J.: 'A 64-b quad-issue CMOS RISC microprocessor', *IEEE J. Solid-State Circuits*, 1996, **31**, pp. 1697–1702
6 GRONOWSKI, P.E.: 'A 433-MHz 64-bit quad-issue RISC microprocessor', *IEEE J. Solid-State Circuits*, 1996, **31**, pp. 1687–1696
7 VASSEGHI, N., YEAGER, K., SARTO, E., and SEDDIGHNEZHAD, M.: '200-Mhz superscalar RISC microprocessor', *IEEE J. Solid-State Circuits*, 1996, **31**, pp. 1675–1686
8 NEVES, J.L., and FRIEDMAN, E.G.: 'Topological design of clock distribution networks based on non-zero clock skew specification'. Proceedings of the 36th IEEE Midwest symposium on *Circuits and systems*, August 1993, pp. 468–471
9 XI, J.G., and DAI, W.W.-M.: 'Useful-skew clock routing with gate sizing for low power design'. Proceedings of the 33rd ACM/IEEE conference on *Design automation*, June 1996, pp. 383–388
10 NEVES, J.L., and FRIEDMAN, E.G.: 'Design methodology for synthesising clock distribution networks exploiting non-zero localised clock skew', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1996, **4**, pp. 286–291
11 JACKSON, M.A.B., SRINIVASAN, A., and KUH, E.S.: 'Clock routing for high-performance ICs'. Proceedings of the 27th ACM/IEEE conference on *Design automation*, June 1990, pp. 573–579
12 TSAY, R.-S.: 'An exact zero-skew clock routing algorithm', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1993, **12**, pp. 242–249
13 CHOU, N.-C., and CHENG, C.-K.: 'On general zero-skew clock net construction', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1995, **3**, pp. 141–146
14 ITO, N., SUGIYAMA, H., and KONNO, T.: 'ChipPRISM: clock routing and timing analysis for high-performance CMOS VLSI chips', *Fujitsu Sci. Tech. J.*, 1995, **31**, pp. 180–187
15 FISHBURN, J.P.: 'Clock skew optimisation', *IEEE Trans. Comput.*, 1990, **39**, pp. 945–951
16 CORMEN, T.H., LEISERSON, C.E., and RIVEST, R.L.: 'Introduction to algorithms' (MIT Press, 1989)
17 WEST, D.B.: 'Introduction to graph theory' (Prentice-Hall, 1996)
18 LEE, T.-C., and KONG, J.: 'The new line in IC design', *IEEE Spectrum*, March 1997, pp. 52–58
19 FRIEDMAN, E.G.: 'The application of localised clock distribution design to improving the performance of retimed sequential circuits'. Proceedings of the IEEE Asia-Pacific conference on *Circuits and systems*, December 1992, pp. 12–17
20 KOURTEV, I.S., and FRIEDMAN, E.G.: 'Simultaneous clock scheduling and buffered clock tree synthesis'. Proceedings of the IEEE international symposium on *Circuits and systems*, June 1997, pp. 1812–1815
21 NEVES, J.L., and FRIEDMAN, E.G.: 'Optimal clock skew scheduling tolerant to process variations'. Proceedings of the ACM/IEEE conference on *Design automation*, June 1996, pp. 623–628
22 DEOKAR, R.B., and SAPATNEKAR, S.S.: 'A graph-theoretic approach to clock skew optimisation'. Proceedings of the IEEE international symposium on *Circuits and systems*, May 1995, pp. 407–410
23 LEISERSON, C.E., and SAXE, J.B.: 'A mixed-integer linear programming problem which is efficiently solvable', *J. Algorithms*, 1988, **9**, pp. 114–128