Clock Skew Scheduling for Improved Reliability via Quadratic Programming*

Ivan S. Kourtev§ and Eby G. Friedman¶

§Department of Electrical Engineering
University of Pittsburgh
Pittsburgh, Pennsylvania 15261
ivan@ee.pitt.edu

¶Department of Electrical and Computer Engineering
University of Rochester
Rochester, New York 14627-0231
friedman@ece.rochester.edu

Abstract— **This paper considers the problem of determining an optimal clock skew schedule for a synchronous VLSI circuit. A novel formulation of clock skew scheduling as a constrained *quadratic programming* (QP) problem is introduced. The concept of a *permissible range*, or a valid interval, for the clock skew of each local data path is key to this QP approach. From a reliability perspective, the ideal clock schedule corresponds to each clock skew within the circuit being at the center of the respective permissible range. However, this ideal clock schedule is *not* practically implementable because of limitations imposed by the connectivity among the registers within the circuit. To evaluate the reliability, a quadratic cost function is introduced as the Euclidean distance between the ideal schedule and a given practically feasible clock schedule. This cost function is the minimization objective of the described algorithms for the solution of the previously mentioned quadratic program. Furthermore, the work described here substantially differs from previous research in that it permits complete control over specific clock signal delays or skews within the circuit. Specifically, the algorithms described here can be employed to obtain results with explicitly specified *target* values of important clock delays/skews with a circuit, such as for example, the clock delays/skews for I/O registers. An additional benefit is a potential reduction in clock period of up to 10%.**

An efficient mathematical algorithm is derived for the solution of the QP problem with $O(r^3)$ run time complexity and $O(r^2)$ storage complexity, where r is the number of registers in the circuit. The algorithm is implemented as a C++ program and demonstrated on the ISCAS'89 suite of benchmark circuits as well as on a number of industrial circuits. The work described here yields additional insights into the correlation between circuit structure and circuit timing by characterizing the degree to which specific signal paths limit the overall performance and reliability of a circuit. This information is directly applicable to logic and architectural synthesis.

I. Introduction

This paper addresses the task of determining a *non-zero* clock skew schedule that satisfies the tighter timing constraints of high speed, VLSI complexity systems. The primary objective of the clock skew scheduling algorithm presented in this paper is to maximize the tolerance of the circuit to process parameter variations. This task is accomplished by first choosing an objective clock skew value for each local data path. A consistent clock skew schedule is determined by application of the optimization algorithm described in this paper. This algorithm minimizes the *least square error* between the computed clock skew schedule and the objective clock skew schedule. As in previous work [1–5], a secondary objective of the clock skew scheduling algorithm is to increase the system-wide clock frequency.

The paper begins with reviewing the circuit graph model in Sec. II. The formulation of the clock skew scheduling problem as a quadratic programming problem is discussed in Sec. III along with the mathematical procedures used to solve the QP problem. Results for ISCAS benchmark and industrial circuits are presented in Sec. IV and some conclusions are briefly offered in Sec. V.

II. Background

In this section, certain properties of a fully synchronous digital system are outlined. Specifically, the timing properties of these systems are described in Sec. II-A and the graph model used to represent these systems is described in Sec. II-B.

A. Timing properties of a synchronous system

An abstract view of a fully synchronous system is shown in Fig. 1. All signal propagation paths within the Combinational Logic are from the input to the output. Alternatively, no purely

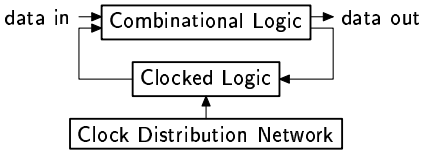


Fig. 1. A fully synchronous system

combinational paths exist through the Clocked Logic—all incoming signals pass through one or more *registers* before reaching the input of the combinational logic. The Clocked Logic does *not* necessarily consist only of discrete registers. Consider a clocked *intellectual property (IP)* block, for example, where the IP internal structure is hidden but each input signal of the IP block is known to latch into a register. With sufficient timing information—(a) the setup and hold times for the latched inputs, and (b) the delays of the paths starting at an internal register and ending at an output—such IP blocks can be viewed as 'generalized registers' and successfully used without exposing the internal circuitry of these blocks.

An example of a *local data path* [5] (a *sequentially-adjacent pair of registers*) delimited by the registers $\mathsf{R_i}$ and $\mathsf{R_f}$ is shown in Fig. 2. Such local data paths are characterized by a minimum and a maximum signal propagation delay from Q_i to D_f. The clock signals C_i and C_f are delivered to $\mathsf{R_i}$ and $\mathsf{R_f}$ with delays t_d^{i} and t_d^{f}, respectively, whereas the algebraic difference, $t_d^{\mathsf{i}} - t_d^{\mathsf{f}}$, is known as the *clock skew* [1, 4, 5]. Note that depending on the path direction and t_d^{i} and t_d^{f}, the clock skew as defined above may be negative, zero, or positive [5].

*This research was supported in part by the National Science Foundation under Grant No. MIP-9423886 and Grant No. MIP-9610108, 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.

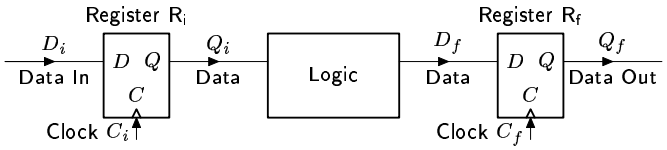


Fig. 2. A local data path

The precise temporal relationships among the C, D, and Q signals in a local data path depend on many factors, including the particular types of registers employed (for an in-depth treatment see [5, 6]). In the majority of cases, however, these timing relationships may be translated into an interval of values which the clock skew may assume [1, 5]. A *permissible range* [7] is associated with each local data path—a clock skew schedule is *feasible* if each local clock skew is within the path specific permissible range. Note that the permissible range of each local data path is guaranteed to include the zero—however, *non*-optimal—clock skew value [8]. Thus, most synchronous circuits are designed to satisfy global zero clock skew. A great deal of effort has been applied to the design of clock distribution networks which maintain global zero clock skew across the circuit [5].

B. Circuit model

The work described in this paper is based upon a *connected undirected graph* [9, 10] model of a synchronous circuit. Formally, the graph $\mathcal{G}_C$ of a circuit C with r registers and p local data paths is the six-tuple $\mathcal{G}_C = \langle V^{(C)}, E^{(C)}, \mathbf{A}^{(C)}, h_l^{(C)}, h_u^{(C)}, h_d^{(C)} \rangle$, where $V^{(C)} = \{v_1, \ldots v_r\}$, $E^{(C)} = \{e_1, \ldots e_p\}$, and $\mathbf{A}^{(C)} = [a_{ij}^{(C)}]_{r \times r}$ are the set of vertices, set of edges, and symmetric adjacency matrix of $\mathcal{G}_C$, respectively. Each vertex from $V^{(C)}$ represents a generalized

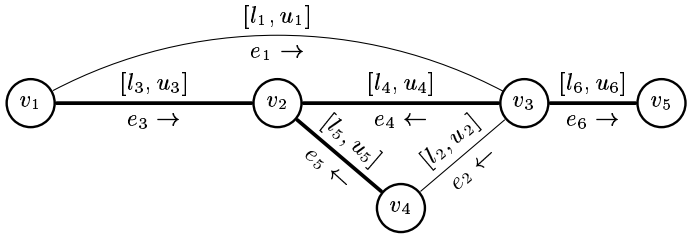
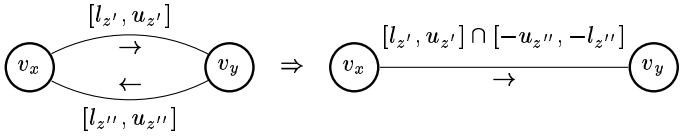


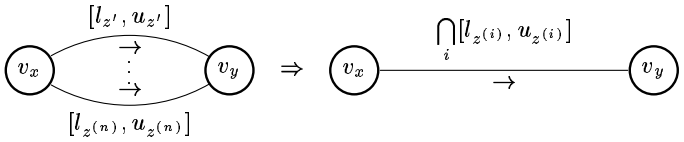
Fig. 3. A circuit graph—edges from the spanning tree are thicker

register of C, *i.e.*, either a discrete register or any block with an individually controlled clock signal. Each edge from $E^{(C)}$ corresponds to a local data path from one generalized register to another. The mappings $h_l^{(C)} : E^{(C)} \mapsto \mathbb{R}$ and $h_u^{(C)} : E^{(C)} \mapsto \mathbb{R}$ to the set of real numbers $\mathbb{R}$ assign the lower and upper clock skew bounds, $l_k, u_k \in \mathbb{R}$, respectively, for $e_k \in E$. The edge labeling $h_d^{(C)}$ defines a direction of signal propagation for each edge v_x, e_z, v_y such that the clock skew $s_z = t_d^{\mathrm{x}} - t_d^{\mathrm{y}}$. For brevity, the superscript (C) is omitted for the rest of the paper unless a circuit must be explicitly indicated, while the terms, register/vertex and edge/local data path, are used interchangeably. A simple example of the graph $\mathcal{G}$ of a circuit with $r = 5$ registers and $p = 6$ local data paths is shown in Fig. 3—note the permissible range $[l, u]$ and the direction arrow labeled on each edge.

A graph is built from a circuit in a natural way (adding a vertex per register and a properly labeled edge per local data path) with two notable exceptions. By construction, $\mathcal{G}$ has no *loops* or *parallel* edges, *i.e.*, $\mathcal{G}$ is a *simple* graph [10]. Loops v_x, e_y, v_x are discarded except for asserting that zero clock skew is within the loop permissible range $[l_y, u_y]$. Multiple edges are eliminated by using the graph transformations depicted in Fig. 4. Specifically,



(a) Graph transformation rule to eliminate two-edge cycles



(b) Graph transformation rule to eliminate multiple edges

Fig. 4. Rules for building a circuit graph

two-edge cycles—such as $v_x, e_{z'}, v_y, e_{z''}, v_x$ in Fig. 4(a)—are collapsed into a *single* edge v_x, e_z, v_y with an arbitrarily chosen direction and the corresponding permissible range (here, the convention is that the edge direction is away from the vertex with the smaller index). Multiple edges in the same direction between two vertices are represented in $\mathcal{G}$ by a single edge as shown in Fig. 4(b)—the edge direction is preserved and the permissible range is such that all permissible ranges are simultaneously satisfied.

III. Clock Skew Scheduling as a QP Problem

The formulation of clock skew scheduling as a *quadratic programming (QP)* problem is described in this section. The linear dependencies among the clock skews and the kernel of cycles are introduced in Sec. III-A. The QP problem is formulated and solved in Sec. III-B. Modifications to the QP problem to handle certain critical practical requirements are analyzed in Sec. III-C.

A. Linear dependence of clock skews

Linear dependencies among the clock skews [8] are illustrated in the circuit shown in Fig. 3 where, for instance, $s_4 = s_2 + s_5$. It is generally possible to identify multiple *minimal* sets of skews such that (a) the skews within the set are *linearly independent*, and (b) every skew within the circuit is a linear combination of skews from the set. A set with these properties is called here a *skew basis*—examples of skew bases in Fig. 3 are $\{s_3, s_4, s_5, s_6\}$ and $\{s_1, s_3, s_5, s_6\}$. Similarly, linear dependencies can be shown among cycles in $\mathcal{G}$—in Fig. 3, the cycle $s_1 + s_2 - s_3 + s_5 = 0$ is a linear combination (the sum in this case) of $s_2 - s_4 + s_5 = 0$ and $s_1 - s_3 + s_4 = 0$. A *kernel* of $\mathcal{G}$ is a minimal set of cycles such that (a) the cycles are linearly independent, and (b) every cycle in $\mathcal{G}$ is a linear combination of cycles from the set. Note that a

skew basis must *not* contain a cycle. If it did, the basis skews would be linearly dependent. Alternatively, linear independence in a kernel is guaranteed by choosing the cycles such that each cycle contains a unique edge from $\mathcal{G}$. From graph theory [9–11], a *spanning tree* of the circuit graph $\mathcal{G}$ defines a basis of exactly $n_b = r - 1$ edges—an example is indicated by the thicker edges shown in Fig. 3. The $n_c = p - n_b$ edges outside the basis are called *chords*—any choice of basis naturally yields a kernel of exactly n_c cycles, each cycle consisting of exactly one chord and the unique path within the basis between the end vertices of this chord. Note that certain basis edges—such as e_6 in Fig. 3—may not belong to any cycle at all (regardless of basis/kernel choice). Such basis edges are called *isolated*, while the rest of the basis edges are called *main*—there are $n_m \leq n_b$ main and $n_i \leq n_b$ isolated basis edges, respectively, where $n_m + n_i = n_b$. Note that the values of the basis skews are sufficient to calculate all skews within $\mathcal{G}$ (basis plus chords). Also, the main basis provides the necessary information to compute the chord skews while the isolated basis can be scheduled to any values—there are no constraints on these edges since these edges do not belong to any cycle.

The kernel can be summarized in a compact way by the *circuit kernel* equation, $\mathbf{Bs} = \mathbf{0}$, where $\mathbf{s}$ is an $n_c + n_m = p'$-element vector of all but the isolated skews, and, each row of the $n_c \times p'$ matrix $\mathbf{B}$ corresponds to a cycle. $\mathbf{B}$ can be derived from inspection by choosing a traversal direction of each cycle and including skews along the cycle with a sign depending upon the edge direction labeling (note the similarity with Kirchoff's Voltage Law loop equations for electrical networks [11]). Assume that the edges/skews are enumerated as in Fig. 3 such that the chords $\mathbf{s}^c$ are first (indices 1 through n_c), followed by the main basis $\mathbf{s}^b$ (indices $n_c + 1$ through p'), and the isolated basis. If the cycles are permuted so as to appear in the order of the chords (*i.e.,* the first row of $\mathbf{B}$ corresponds to e_1/s_1, and so on), the kernel equation is $\mathbf{Bs} = [\mathbf{I}_{n_c}\ \mathbf{C}_{n_c \times n_m}] \begin{bmatrix} \mathbf{s}^c \\ \mathbf{s}^b \end{bmatrix} = \mathbf{s}^c + \mathbf{C}\mathbf{s}^b = \mathbf{0}$, where $\mathbf{I}_{n_c}$ is an identity matrix of dimension n_c. The solutions of this equation comprise the *kernel* or *null space* $\ker(\mathbf{B})$ of the linear mapping $\mathbf{B} : \mathbb{R}^{p'} \mapsto \mathbb{R}^{n_c}$ and $\mathbf{s}$ is called *consistent* if $\mathbf{s} \in \ker(\mathbf{B})$ [12].

B. QP clock skew scheduling problem formulation and solution

Let an *objective* clock schedule $\mathbf{g}$ be chosen according to certain design criteria ($\mathbf{g}$ has p' elements). From a reliability perspective, for example, an ideal, although most likely *not* consistent, choice of $\mathbf{g}$ is $g_i = (l_i + u_i)/2$. The optimization goal is to determine a feasible and consistent schedule $\mathbf{s}$ such that the least square error $\varepsilon = (\mathbf{s} - \mathbf{g})^2$ is minimized:

$$\begin{aligned} \min \quad & \varepsilon = (\mathbf{s} - \mathbf{g})^2 = \textstyle\sum_{k=1}^{p'} (s_k - g_k)^2 \\ \text{subject to} \quad & \mathbf{Bs} = \mathbf{0} \text{ and } s_k \in [l_k, u_k] \text{ for } k \in \{1 \ldots p'\}. \end{aligned} \tag{1}$$

The problem described by (1) is a constrained QP problem with bounded variables—methods such as *active constraints* exist for solving such problems [13–17]. These methods are both analytically and numerically challenging so a *two-phase* solution process is suggested here such that a constrained version of problem (1) is solved initially. If the result is not feasible, a rapidly converging iterative refinement of the objective $\mathbf{g}$ is performed until the feasibility of $\mathbf{s}$ is satisfied. Expanding the ε term in (1) yields

$$\begin{aligned} \text{Phase 1} \quad \rightarrow \quad & \min \quad \varepsilon = \mathbf{s}^t\mathbf{s} - 2\mathbf{g}^t\mathbf{s} + \mathbf{g}^t\mathbf{g} \\ & \text{subject to} \ \ \mathbf{Bs} = \mathbf{0} \\ \text{Phase 2} \quad \rightarrow \quad & \text{Iterative refinement of } \mathbf{s}, \end{aligned} \tag{2}$$

where Phase 1 is solved using the classical method of *Lagrange multipliers* [16]. If $\mathbf{m}$ is the Lagrange multipliers vector, the *Lagrangian* function is $\mathcal{L}(\mathbf{s}, \mathbf{m}) = \mathbf{s}^t\mathbf{s} - 2\mathbf{g}^t\mathbf{s} + \mathbf{m}^t\mathbf{Bs}$, where the numerical constant $\mathbf{g}^t\mathbf{g}$ from (2) can be omitted without affecting the solution. Any extrema of ε necessarily is a *stationary* point of the Lagrangian, *i.e.,* a solution of the *linear* system,

$$\left| \begin{matrix} 2\mathbf{s} + \mathbf{B}^t\mathbf{m} = 2\mathbf{g} \\ \mathbf{Bs} = \mathbf{0} \end{matrix} \right. \quad \Rightarrow \quad \begin{bmatrix} 2\mathbf{I}_{p'} & \mathbf{B}^t \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ \mathbf{m} \end{bmatrix} = 2 \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix}. \tag{3}$$

Since $\mathbf{B}$ is in row echelon form, $\ker(\mathbf{B}^t) = \{\mathbf{0}\}$ and the matrix $\mathbf{M} = \mathbf{B}\mathbf{B}^t$ is *positive-definite* and *nonsingular*. Thus, the unique solution of (3) by Gaussian elimination is

$$\mathbf{m}^* = 2\mathbf{M}^{-1}\mathbf{B}\mathbf{g}, \ \ \mathbf{s}^* = \mathbf{g} - \frac{1}{2}\mathbf{B}^t\mathbf{m}^* = (\mathbf{I}_{p'} - \mathbf{B}^t\mathbf{M}^{-1}\mathbf{B})\mathbf{g}, \tag{4}$$

where $\mathbf{s}^*$ in (4) is also a global minimizer of ε in (2) [17].

The size and inversion of $\mathbf{M}$ can make computing (4) a significant challenge, thereby prompting the need for an efficient strategy to calculate $\mathbf{s}^*$. Specifically, $\mathbf{M}^{-1} = \mathbf{I}_{p'} - \mathbf{C}\mathbf{N}^{-1}\mathbf{C}^t$, where $\mathbf{N} = \mathbf{I}_{n_m} + \mathbf{C}^t\mathbf{C}$ (Sherman-Morrison-Woodburry formula [18]) and similarly to $\mathbf{M}$, can be argued to be necessarily nonsingular. The dimensions of $\mathbf{N}$ and $\mathbf{M}$ are n_m and n_c, respectively—n_m is typically an order of magnitude smaller than n_c resulting in significant savings of time and memory. The basis $\mathbf{s}^{b*}$ is

$$\mathbf{s}^{b*} = [-(\mathbf{L}_2^{-t}\mathbf{L}_2^{-1})\mathbf{C}^t\ \mathbf{L}_2^{-t}\mathbf{L}_2^{-1}]\mathbf{g}, \tag{5}$$

where $\mathbf{L}_2$ is the *Cholesky* decomposition [18–20] of $\mathbf{N}$. The chords $\mathbf{s}^{c*}$ are determined from the kernel equation $\mathbf{s}^{c*} = -\mathbf{C}\mathbf{s}^{b*}$, where the multiplication by $\mathbf{C}$ can be carried out extremely fast since the elements of $\mathbf{C}$ are either -1, 0, or 1. Finally, the iterative refinement of the objective clock schedule $\mathbf{g}$ in Phase 2 is accomplished as follows. For lower and upper bound violations, move the objective clock skew halfway between its current value and the closer of *zero* and the lower and upper bounds, respectively, of the permissible range. For paths with no violation, move the objective clock skew halfway between the current value and the actual value obtained by (5).

The computation in (5) requires $\frac{1}{6}n_m^3$, $\frac{1}{6}n_m^3 + \frac{1}{2}n_m^2 + \frac{1}{3}n_m$, $\frac{1}{6}n_m^3 + \frac{1}{6}(5n_m^2 + n_m - 1)$, and $n_m p$ multiplications to determine $\mathbf{L}_2$, $\mathbf{L}_2^{-1}$, the product $\mathbf{L}_2^{-t}\mathbf{L}_2^{-1}$, and $\mathbf{s}^{b*}$, respectively. A maximum of n_m^2 floating point storage units is used during this procedure. $\mathbf{N}$, $\mathbf{L}_2$, and $\mathbf{L}_2^{-1}$ are all symmetric (only half of the elements must be stored) and can be computed 'in place,' *i.e.,* $\mathbf{L}_2$ and $\mathbf{L}_2^{-1}$ replace $\mathbf{N}$ and $\mathbf{L}_2$, respectively, as the matrices are computed. Therefore (n_m is of the order of r), the run time and memory complexity of the described algorithm are $O(r^3)$ and $O(r^2)$, respectively.

C. Enhancing clock skew scheduling to satisfy target delays

An important object of clock skew scheduling is to control certain clock signal delays. Consider, for example, the i/o registers

of a chip. Given the difficulty in knowing *a priori* all timing contexts of an integrated circuit, a preferred solution may be to require that all i/o registers are clocked at the same time (zero skew). All explicit delay requirements fall into one of the following categories
1. *zero skew island, i.e.,* a group of registers with equal delay,
2. target delays, *i.e.,* $t_d^{k1} = \delta_{k_1}, \ldots, t_d^{k\alpha} = \delta_{k_\alpha}$, $k_\alpha \leq r$,
3. target skews, *i.e.,* $s_{j_1} = \sigma_{j_1}, \ldots, s_{j_\beta} = \sigma_{j_\beta}$, $j_\beta < n_b$.

Zero skew islands can be satisfied by collapsing the corresponding graph vertices into a *single* vertex while eliminating all edges among vertices from the island. Note that zero skew is within the permissible range of each in-island path. Alternatively, the target delays are converted to target skews (category 3 above) for sequentially-adjacent pairs or by adding a 'fake' edge. Thus, an algorithm to handle only target skews is necessary.

Note first that target values for only $n_f \leq n_b$ skews can be independently specified. As n_f approaches n_b, the freedom to vary all skews decreases and it may become impossible to determine any feasible $\mathbf{s}$. Given $n_f \leq n_b$, (a) the basis can always be chosen to contain all target skews by using a spanning tree algorithm with edge swapping, and (b) the edge enumeration can be accomplished such that the target skews appear last in the basis. The problem is now similar to (2) except for the change of the circuit kernel equation,

$$\mathbf{C} = [\mathbf{C}_1\ \mathbf{C}_2],\ \mathbf{Bs} = [\mathbf{I}\ \mathbf{C}_1\ \mathbf{C}_2] \begin{bmatrix} \hat{\mathbf{s}}^c \\ \hat{\mathbf{s}}^b \\ \sigma \end{bmatrix} \Rightarrow \hat{\mathbf{B}}\hat{\mathbf{s}} + \mathbf{C}_2\sigma = \mathbf{0}, \qquad (6)$$

where $\hat{\mathbf{B}} = [\mathbf{I}\ \mathbf{C}_1]$, $\hat{\mathbf{s}} = \begin{bmatrix} \hat{\mathbf{s}}^c \\ \hat{\mathbf{s}}^b \end{bmatrix}$, $\hat{\mathbf{s}}^c = \mathbf{s}^c$, and $\hat{\mathbf{s}}^b$ is $\mathbf{s}^c$ with the last n_f elements removed. The matrix $\mathbf{C}_2$ in (6) consists of the last n_f columns of $\mathbf{C}$, while the target skew vector σ is an n_f-element vector of target skews whose elements are ordered in the order of the target edges. The linear system (3) becomes

$$\begin{vmatrix} 2\hat{\mathbf{s}} + \hat{\mathbf{B}}^t\hat{\mathbf{m}} = 2\hat{\mathbf{g}} \\ \hat{\mathbf{B}}\hat{\mathbf{s}} + \mathbf{C}_2\sigma = \mathbf{0} \end{vmatrix} \Rightarrow \begin{bmatrix} 2\mathbf{I} & \hat{\mathbf{B}}^t \\ \hat{\mathbf{B}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{s}} \\ \hat{\mathbf{m}} \end{bmatrix} = \begin{bmatrix} 2\hat{\mathbf{g}} \\ -\mathbf{C}_2\sigma \end{bmatrix}, \qquad (7)$$

with solution

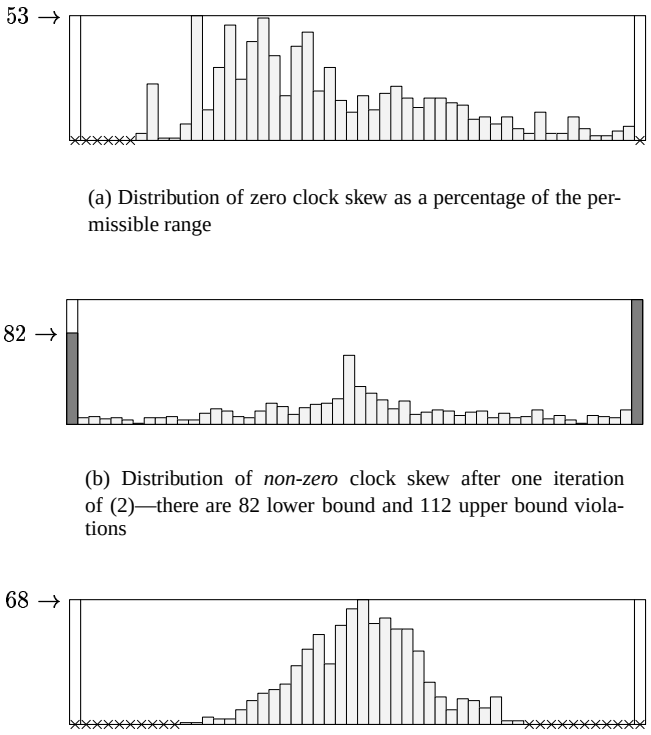
$$\begin{aligned} \hat{\mathbf{m}}^* &= 2\hat{\mathbf{M}}^{-1}(\hat{\mathbf{B}}\hat{\mathbf{g}} + \mathbf{C}_2\sigma) \\ \hat{\mathbf{s}}^* &= (\mathbf{I} - \hat{\mathbf{B}}^t\hat{\mathbf{M}}^{-1}\hat{\mathbf{B}})\hat{\mathbf{g}} - \hat{\mathbf{B}}^t\hat{\mathbf{M}}^{-1}\mathbf{C}_2\sigma. \end{aligned} \qquad (8)$$

IV. Results

The algorithm described in Sec. III-B has been implemented as a `C++` program and applied to both the ISCAS'89 benchmark circuits and some industrial circuits (`IC1, IC2` and `IC3`). The results of the application of the algorithm to these circuits are summarized in Table I. For each circuit, the following data is listed—the circuit name in column **1**, the number of disjoint subgraphs in column **2**, and the number of vertices, edges, chords (cycles), main and isolated basis, and target clock period in nanoseconds in columns **3** through **8**, respectively. The number of iterations to reach a solution is listed in column **9**. The average value of ε in (2), *i.e.,* $\sqrt{\varepsilon/p}$ is listed in column **10**. The run time in minutes for the mathematical portion of the program is shown in column **11** for a 170 MHz Sun Ultra 1 workstation. For circuits with more than one disjoint subgraph, the algorithm is applied to each subgraph separately.

The results are illustrated in Fig. 5 where a histogram of the clock skew distribution within the permissible range is graphically displayed for the specific circuit `s3271`. After one iteration of (2) with all objective clock skews set at the center of the permissible range, there are 82 lower bound and 112 upper bound violations, respectively [shown darker in Fig. 5(b)]. At the conclusion of execution—depicted in Fig. 5(c)—no violations remain. Also shown in Fig. 5(a) is the distribution of zero clock skew. Note the difference between Fig. 5(a) and Fig. 5(c). There is a clear improvement in that any non-zero clock skew is at least 18% from either edge of the corresponding permissible range. The majority of the non-zero clock skews are within 5% of the safest clock skew value at the center of the permissible range.

V. Conclusions



(a) Distribution of zero clock skew as a percentage of the permissible range

(b) Distribution of *non-zero* clock skew after one iteration of (2)—there are 82 lower bound and 112 upper bound violations

(c) Distribution of *non-zero* clock skew after all three iterations of (2)—there are no lower or upper bound violations

Fig. 5. The distribution of clock skew within the permissible range (as a percentage from 0% to 100%) for the circuit `s3271` (note that height is not drawn to scale). Intervals that are crossed out contain no skews.

The problem of clock skew scheduling for improved tolerance to process parameter variations is examined in this paper. A novel formulation of the optimal scheduling problem from a reliability perspective is presented as a Quadratic Programming (QP) problem. In this approach,
- the objective clock period is fixed,
- the objective clock skews for each local data path are chosen according to any specific design criteria,
- any requirements for explicit specification of clock signal delays within a circuit (target skews) are fully supported,

TABLE I

EXPERIMENTAL RESULTS

Circuit	# subcircuits	r	p	n_c	n_m	n_i	T_{CP}	# iterations	$\sqrt{\frac{\varepsilon}{p}}$	Run time (min)	Circuit	# subcircuits	r	p	n_c	n_m	n_i	T_{CP}	# iterations	$\sqrt{\frac{\varepsilon}{p}}$	Run time (min)
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
s1196	7	18	20	9	8	3	20.8	5	3.19	1	s526n	1	21	117	97	20	0	13	2	1.26	2
s1238	7	18	20	9	8	3	20.8	5	3.19	1	s5378	1	179	1147	969	158	20	28.4	20	8.79	3
s13207	49	669	3068	2448	581	39	85.6	20	18.92	5	s641	1	19	81	63	18	0	83.6	5	11.67	1
s1423	2	74	1471	1399	72	0	92.2	20	60.9	3	s713	1	19	81	63	18	0	89.2	6	12.74	1
s1488	1	6	15	10	5	0	32.2	1	0.87	1	s820	1	5	10	6	4	0	18.6	1	0.71	1
s1494	1	6	15	10	5	0	32.8	1	0.88	1	s832	1	5	10	6	4	0	19	2	0.66	1
s15850	15	597	14257	13675	546	36	116	10	70.6	21	s838.1	1	32	496	465	31	0	24.4	3	3.68	3
s15850.1	22	534	10830	10318	478	34	81.2	9	31.44	19	s9234	3	228	2476	2251	222	3	75.8	20	16.67	4
s208.1	1	8	28	21	7	0	12.4	1	1.22	1	s9234.1	2	211	2342	2133	205	4	75.8	20	18.6	4
s27	1	3	3	1	2	0	6.6	1	0.71	1	s953	4	29	135	110	25	0	23.2	3	1.93	2
s298	1	14	54	41	12	1	13	1	1.16	1	s1269	1	37	251	215	36	0	51.2	20	12.73	2
s344	1	15	68	54	14	0	27	4	4.91	1	s1512	1	57	405	349	56	0	39.6	4	4.43	3
s349	1	15	68	54	14	0	27	4	4.91	1	s3271	1	116	789	674	107	8	40.4	3	3.64	5
s35932	1	1728	4187	2460	1727	0	34.2	20	60.4	27	s3330	1	132	514	383	61	70	34.8	4	3.4	5
s382	1	21	113	93	20	0	14.2	6	1.59	2	s3384	25	183	1759	1601	151	7	85.2	5	15.5	7
s38417	11	1636	28082	26457	1443	182	69	20	32.35	31	s4863	1	104	620	517	103	0	81.2	8	39.85	3
s38584	2	1452	15545	14095	1400	50	94.2	11	29.1	29	s6669	20	239	2138	1919	218	1	128.6	3	20.67	6
s386	1	6	15	10	5	0	17.8	1	0.82	1	s938	1	32	496	465	31	0	24.4	2	3.41	2
s400	1	21	113	93	20	0	14.2	8	1.6	1	s967	4	29	135	110	25	0	20.6	2	1.76	2
s420.1	1	16	120	105	15	0	16.4	20	1.95	1	s991	1	19	51	33	18	0	96.4	3	8.58	1
s444	1	21	113	93	20	0	16.8	2	1.05	1	IC1	1	500	124750	124251	499	0	8.2	2	1.51	30
s510	1	6	15	10	5	0	16.8	1	0.85	1	IC2	1	59	493	435	58	0	10.3	3	1.82	4
s526	1	21	117	97	20	0	13	2	1.26	1	IC3	34	1248	4322	3108	1155	59	5.6	2	1.43	2

- the least square distance between the objective and actual values of the clock skew vector over the entire circuit is minimized.

An efficient computational procedure is introduced to solve the QP problem via the equality constrained Lagrange Multipliers method with iterative refinement of the objective. This procedure—with $O(r^3)$ run time complexity and $O(r^2)$ memory complexity, respectively—has been implemented and demonstrated on a number of benchmark and industrial circuits.

REFERENCES

[1] J. P. Fishburn, "Clock Skew Optimization," *IEEE Transactions on Computers*, Vol. C–39, pp. 945–951, July 1990.

[2] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Graph Algorithms for Clock Schedule Optimization," *Proceedings of the IEEE/ACM International Conference on Computer–Aided Design*, pp. 132–136, November 1992.

[3] 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*, pp. 407–410, May 1995.

[4] T. G. Szymanski, "Computing Optimal Clock Schedules," *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 399–404, June 1992.

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

[6] S. H. Unger and C.-J. Tan, "Clocking Schemes for High-Speed Digital Systems," *IEEE Transactions on Computers*, Vol. C-35, pp. 880–895, October 1986.

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

[8] I. S. Kourtev and E. G. Friedman, "Topological Synthesis of Clock Trees for VLSI-Based DSP Systems," *Proceedings of the IEEE Workshop on Signal Processing Systems*, pp. 151–162, November 1997.

[9] E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice.* Prentice-Hall, 1977.

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

[11] S.-P. Chan, S.-Y. Chan, and S.-G. Chan, *Analysis of Linear Networks and Systems : A Matrix-Oriented Approach with Computer Applications.* Addison-Wesley Publishing Company, 1972.

[12] O. Bretscher, *Linear Algebra with Applications.* Prentice-Hall, 1996.

[13] P. G. Ciarlet and J. L. Lions, eds., *Handbook of Numerical Analysis,* Vol. I. North-Holland, 1990.

[14] R. W. Farebrother, *Linear Least Square Computations.* Marcel Dekker, 1988.

[15] M. R. Osborne, *Finite Algorithms in Optimization and Data Analysis.* John Wiley & Sons, 1985.

[16] R. Fletcher, *Practical Methods of Optimization.* John Wiley & Sons, 1987.

[17] Å. Björck, *Numerical Methods for Least Squares Problems.* North-Holland, 1996.

[18] G. H. Golub and C. F. V. Loan, *Matrix Computations.* Johns Hopkins University Press, 1996.

[19] G. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems.* Prentice-Hall, 1967.

[20] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems.* Prentice-Hall, 1974.