Integration of Clock Skew and Register Delays into a Retiming Algorithm

Tolga Soyata, Eby G. Friedman, and J. H. Mulligan, Jr.*

University of Rochester
Department of Electrical Engineering
Rochester, New York 14627

*University of California
Department of Electrical and Computer Engineering
Irvine, California 92717

***Abstract* — The clock frequency of a synchronous circuit can be increased by retiming, an operation of temporally and physically relocating the registers. In this paper, a new approach to the retiming process is presented which enables one to consider the effects on optimal retiming of electrical issues such as variable clock distribution delays and different register delays due to variable loads and cell instances. The algorithm provides increased accuracy in determining the maximum clock frequency and also eliminates any race conditions. Depending on the nature of the synchronous circuit, retiming using this algorithm may also provide an increase in system operating clock frequency.**

I. Introduction

In synchronous circuits, digital signals are stored in registers awaiting a synchronizing clock pulse. Each register delays the signal by a single clock period. The operation of placing registers within a combinatorial circuit so as to increase the clock frequency of the circuit is known as **pipelining** [1–5]. Optimization of the pipelined synchronous circuit by relocating these registers in order to achieve an increased clock frequency is known as **retiming** and was initially developed by Leiserson and Saxe [6, 7] in 1983.

Early work in the field of retiming assumes registers with no delay and zero or negligible clock skew [6, 7]. De Micheli [8] considers the retiming problem without separating the combinational components from the registers. In his work, he shows that if the set-up time (t_s) and clock-to-Q time (t_p) are not zero, a reduced effective cycle time of $T-t_s-t_p$ must be considered. This implicitly assumes equal register delays. In [9], Malik *et al.* integrates retiming with logic synthesis by temporarily removing registers from the circuit so as to do combinational optimization on the logic elements. This work also assumes negligible clock skew and register delays. In [10], Lockyear and Ebeling study optimal retiming using level-clocked latches. Their work is based on permitting level-sensitive latches to borrow cycle time across adjacent latches that would be lost if edge-clocked registers were used. More recently, Szymanski [11] investigates optimal multiphase clocking and timing verification and provides more selective constraint equations; thereby improving the computational efficiency, although he does not apply these concepts to retiming. Burks, Sakallah, and Mudge investigate retiming using multiphase clocking [12] and do not consider clock skew and variable register delays. In [13], Sakallah and his colleagues address the retiming problem in terms of the optimal clocking of synchronous circuits using linear programming techniques. They provide benchmark examples of improved performance which can be obtained as a result.

Clock skew can seriously affect the performance of a synchronous system and must be considered during the retiming process. Clock skew is manifested by a lead/lag relationship between two sequentially adjacent registers [14–18]. In addition, certain clock skew lead/lag relationships can be used to improve performance. This attribute has been shown to improve maximum clock rates by up to 30%, depending upon the nature of the circuit architecture [14, 15]. Undesirable clock skew introduced as a result of retiming can produce a net negative delay for a local data path. This implies the existence of a race condition, which must be avoided as a condition imposed on the retiming process. Thus, the algorithms proposed in this paper can be used to improve the clock frequency and to ensure that no race conditions are created while retiming. A retiming algorithm that has no provision for including variable register delays with suitable capability for incorporating variations in cell instances and variable loads can be expected to limit the accuracy of the estimated improvements in clock frequency. Depending upon the relative magnitude of the register and clock delays, the value of the predicted clock frequency after retiming may be less than the original circuit.

In this paper, both clock skew and variable register delays have been integrated into the retiming algorithm presented in this paper. To accomplish this result, a path between logic elements is defined in this paper as the traversal from weighted edge to weighted edge, the latter being interpreted as a connection containing one or more registers between logic blocks. With this definition, clock and register delays can then be assigned to each edge. Thus, as registers are shifted from edge to edge, different clock skews and register delays are developed and considered in each of the path delays. This permits both clock period limitations and race conditions to be easily detected on a path-by-path basis. Estimates of register delays on zero weight edges (i.e., no registers) are required in order to include the effects of variable register delays on the retimed circuit. This approach, therefore, initially requires approximate (or expected) values of register and clock delays which can be replaced with more accurate values as the exploratory pipelining process becomes better specified. This structure provides the basis for a polynomial-time retiming algorithm which determines the register locations which achieve a maximum operating clock frequency for the system and freedom from race conditions. This is the primary result of this paper.

In section II, the effects of non-zero clock skew and unequal register delays are quantified for their use in the retiming algorithms. In section III, both the original Leiserson-Saxe algorithm and the proposed algorithm are summarized and their differences outlined. Details of the algorithm development are described in section IV. Examples of the performance of some circuits retimed with this algorithm are described in section V. Finally, in section VI, some conclusions are presented.

II. Quantification of Register and Clock Distribution Characteristics

To achieve the principal objectives of the new algorithm, a number-triple, the **Register Electrical Characteristic** (REC), is assigned to each register in the form of $T_{CD} : T_{SET-UP}/T_{C\rightarrow Q}$. T_{CD} is the clock delay from the clock source to each register, T_{SET-UP} is the time required for the data at the input of a register to latch, and $T_{C\rightarrow Q}$ is the time required for the data to appear at the output of the register upon arrival of the clock signal.

Let $T_{CD}(i) : T_{SET-UP}(i)/T_{C\rightarrow Q}(i)$ represent the REC for the register located on edge i. Then, the **clock skew**, T_{SKEW}, between two sequentially adjacent registers i and j is defined as

$$T_{SKEW}(i,j) = T_{CD}(i) - T_{CD}(j) \tag{1}$$

If $T_{CD}(j) > T_{CD}(i)$, the clock skew is defined to be negative. If $T_{CD}(j) < T_{CD}(i)$, the clock skew between these registers is positive. In the case that $T_{CD}(j)$ equals $T_{CD}(i)$, i.e., the clock signal reaches the clock input of the two registers at exactly the same time, the clock skew is defined to be zero.

Two sequentially adjacent registers R_i and R_j form a local data path. If logic elements exist between R_i and R_j, the local data path delay $T_{PD}(i,j)$ is

$$T_{PD}(i,j) = T_{C\rightarrow Q}(i) + T_{LOGIC}(i,j) + T_{SET-UP}(j) + T_{SKEW}(i,j) \tag{2}$$

where $T_{LOGIC}(i,j)$ is the total delay of the logic elements and interconnect between these registers. If there are parallel paths between the two registers, there is a $T_{LOGIC}(i,j)$ for each path.

This research is based upon work supported by the National Science Foundation under Grant No. MIP-9208165

III. Overview of Retiming Algorithms

Given an architecture consisting of logic blocks and registers which is capable of operating at some maximum clock frequency, the object of retiming is to relocate the registers to achieve an increase of maximum operating clock frequency, leaving the latency of the retimed system unchanged from the original. In this section, the Leiserson-Saxe algorithm is outlined briefly and the proposed retiming algorithm is described in conceptual terms.

A. Leiserson-Saxe Algorithm

In [7], a graph-oriented method is used to represent a synchronous circuit. In this method, vertices represent logic elements and edges represent the connectivity and direction of the data flow. A weight, which is equal to the number of registers between two logic blocks, is assigned to each edge. In this algorithm, each time a register is moved across a logic block, a lag of one clock period is added (subtracted) to (from) that vertex. Thus, as the number of registers (the weights) are shifted from edge to edge, the lags attached to each vertex change accordingly, to account for the temporal shift at each vertex. This process is iterated until a minimum clock period is determined or the search space is exhausted.

Fig. 1 is a graph of a digital correlator which was presented by Leiserson and Saxe [7]. In this graph, there are eight vertices and 11 edges. Only four of the 11 edges (e_0, e_1, e_2, and e_3) have registers on them. The graph of Fig. 2 depicts a retimed version of the graph of Fig. 1. The registers on e_0 and e_2 have been removed while registers have been added to e_8 and e_9. Also note that since a register has been removed from e_0, an additional register must be added to e_7 to maintain the latency of the original $e_0 - e_7 - e10$ path. A more complete explanation of both the algorithm and these graphs is provided in [7].

B. Proposed Algorithm

A modified version of the graph of Fig. 1 is shown in Fig. 3 in which a REC is assigned to each edge. By assigning a clock delay to each edge, the circuit area is assumed to be partitioned into regions of similar clock delay, i.e., registers that are located on the same edge are physically located in the same clock delay region. Therefore, registers that end up on the same edge after retiming are assumed to have similar clock delays. Registers that move to different edges are assumed to have the clock and register delays of the new edge. Since registers on different edges may be considered to have different clock and register related delays, moving a register from one edge to another edge during retiming can not only create different data paths with different logic and register delays, but also can change the clock skew of the new local data paths.

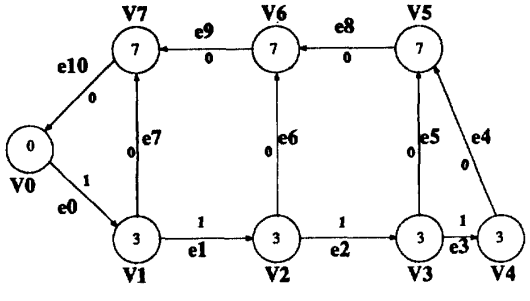

Fig. 1: Graph of digital correlator [7]

In this algorithm, each edge and vertex is represented by a set of logical minterms with delay coefficients. This system of minterms represents a set of possible path delays between possible register locations. These sets of logical minterms are evaluated to determine a more constrained set of minterms which define a set of register locations and which satisfy a target clock period requirement. If the target clock period can be attained, the same system of logical minterms is applied to a smaller clock period. The process is continued in order to derive the minimum possible clock period. This process establishes a final choice of register locations which provides the minimum clock period.

IV. Details of the Proposed Algorithm

A. The L Matrix

The L matrix is a square matrix whose element $L(i,j) = T_{PD}(i,j)$ is the path delay from R_i to R_j. Since the graph is formed by directional edges, $L(i,j)$ is not necessarily equal to $L(j,i)$. Each matrix element contains the summation of the following terms: the delays of registers R_i and R_j, the delays of the logic and interconnect between the registers, and the clock skew between R_i and R_j. This is the quantity $T_{PD}(i,j)$ defined by (2). The size of the matrix is $R \times R$, where R is the number of edges with a positive weight. The maximum value of R is equal to the number of edges E. The value of R can increase or decrease as a result of the retiming process.

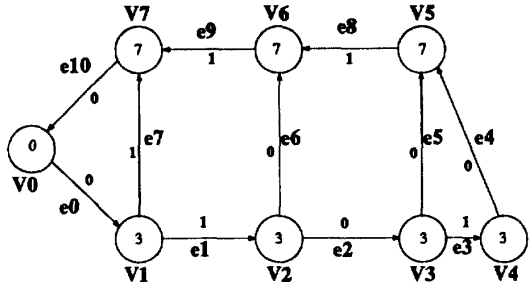

Fig. 2: Retimed version of the graph of Fig. 1

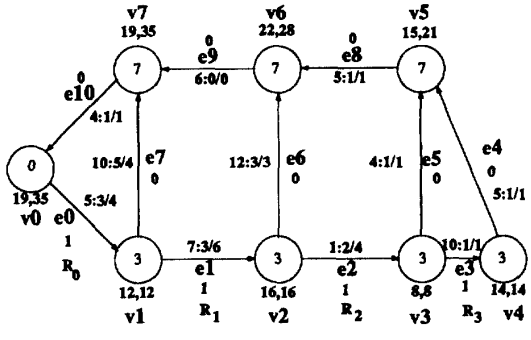

Fig. 3: Graph of Fig. 1 with the addition of REC values

$$L = \begin{array}{c} \\ R_0 \\ R_1 \\ R_2 \\ R_3 \end{array} \begin{array}{c} \begin{array}{cccc} R_0 & R_1 & R_2 & R_3 \end{array} \\ \begin{bmatrix} 17 & 8 & x & x \\ 28 & x & 17 & x \\ 27 & x & x & -1 \\ 33 & x & x & x \end{bmatrix} \end{array}$$

Fig. 4: The L matrix of the graph of Fig. 3

The importance of the L matrix is that the element having the greatest value is the reciprocal of the maximum clock frequency. The process of successfully retiming can be viewed as producing a transformed L matrix, simultaneously achieving a maximum element value which is less than the maximum entry in the original L matrix without incurring any negative entry. The former condition achieves an improvement of the maximum clock frequency, whereas the latter ensures that no race condition exists.

In the event that two or more parallel paths exist between R_i and R_j, the concept of the L matrix must be extended to two matrices, L_{WCP} and L_{BCP}. The elements of these two L matrices are, respectively, $T_{WCPD}(i,j)$ and $T_{BCPD}(i,j)$, which are the longest and the shortest delays of the set of parallel local data paths.

The L matrix of the graph of Fig. 3 is shown in Fig. 4, where "x" denotes the absence of a path between registers R_i and R_j. Note that in Fig. 3, there are no parallel paths. Note also that the negative entry in row 3 and column 4 in Fig. 4 indicates the existence of a race condition.

B. Algorithm CPL

The local path delay from R_i to R_j involves passing through one or more vertex v. References to (1) and (2) indicate that $T_{PD}(i,j)$, the path delay element located in row i and column j of the L matrix, can be grouped into three categories. $T_{CD}(i)$ and $T_{C\rightarrow Q}(i)$ are associated with R_i. T_{SET-UP} and $-T_{CD}(j)$ are associated with R_j. $T_{LOGIC}(i,j)$ is associated with the logic vertices between R_i and R_j. The difference in the values of $T_{PD}(i,j)$, $T_{PD}(i,k)$, and $T_{PD}(i,m)$ represent different local data paths which all start at R_i and share a common vertex v, and, possibly, have separate logic vertices. These paths terminate at registers, R_j, R_k, and R_m, respectively, and the path delays include the terms $T_{SET-UP}(j)$ and $-T_{CD}(j)$ or the corresponding terms with j replaced by k or m, respectively. This fact provides the basis for Algorithm CPL. At each vertex v, a calculation is made of the cumulative delay from R_i to that vertex, $\Delta(v)$. If parallel paths exist, $\Delta(v)$ is defined as the maximum value of the sums of the delays to that vertex v. If a vertex is included in more than one local data path, a worst case delay $\Delta_W(v)$ and best case delay $\Delta_B(v)$ are determined for that vertex. At the end of each local data path, the set-up time of the final register is added while the final clock delay is subtracted to both vertex delays ($\Delta_W(v)$ and $\Delta_B(v)$). These delays are then searched for the maximum value and for any negative entry.

Note that in Fig. 3 each vertex has two delays attached to it. It can be observed from the figure that the maximum logic path delay occurs between registers R_3 and R_0 since Δ_W of v_0 (the vertex that is connected behind R_0) has a Δ_W of 35; therefore, the worst case path delay of $T_{PD}(R_3,R_0) = 35+3-5 = 33$, the general form of which is shown in (3).

$$T_{PD_{MAX}} = \Delta_W + T_{SET-UP}(j) - T_{CD}(j) \qquad (3)$$

The worst case path delay of the circuit is the path delay determined by the path $R_3 \rightarrow v4 \rightarrow v5 \rightarrow v6 \rightarrow v7 \rightarrow v0 \rightarrow R_0$. Note the race condition along the path from R_2 to R_3 with a path delay of $8+1-10 = -1$, which can also be observed in the L matrix element $L(3,4)$ in Fig. 4.

C. Algorithm POLY for Retiming

Retiming is successfully performed by transforming the L matrix into a new L matrix containing a smaller minimum clock period. Creating a functionally equivalent matrix to the original L matrix is performed by changing an edge weight (i.e., the number of registers along an edge), and then changing all other weights accordingly. Note that if two graphs represent functionally equivalent circuits, then every vertex receives an input signal with the same amount of delay, i.e., if a signal at an input of a vertex is delayed by one clock period, then all other input signals at this vertex must be delayed by the same number of clock periods.

A polynomial-time retiming algorithm is introduced in this section which exploits both nonlinear boolean algebra minimization [19] and linear programming [20]. In this algorithm, a boolean variable is associated with the weight of each edge. If the edge has a positive weight, its associated boolean variable is defined to be logic "1," whereas if the edge weight is zero, the associated boolean variable is logic "0." A boolean value of x denotes a "don't care" and can be either 0 or 1.

A sum of logical minterms is formed for each possible register-to-register path in which the boolean "0" state is used to designate an edge without a register and the "1" state, an edge with a register. Thus, each path is described as a sum of minterms, each minterm representing a choice for inserting a register along the path. Each individualized minterm is also multiplied by a delay value, as shown in (4),

$$\begin{aligned} Q_W(v5) &= 40c'b + 29c'b'a' + 28c + 26c'b'a \\ Q_B(v5) &= 19a' + 16a \end{aligned} \qquad (4)$$

where Q is the minterm equation which describes the delay and register conditions and the number designates the delay to that vertex under different register placement conditions. The subscript W and B represent worst case and best case delays. Once these Q terms are calculated for each vertex and edge, the placement of the registers are determined so that a specified clock period is achieved. After the boolean edge weights are defined, linear programming methods, such as Bellman-Ford equations [20], are used to determine the location of the remaining registers. Upon determining the minimum clock period with no race conditions, additional registers may be added to maintain the latency attributes of the original system.

V. Example Circuits

The algorithms CPL and POLY are implemented in C on a SUN 4 workstation. Table 1 shows the results obtained from retiming five example circuits. The corresponding graphs are shown in Figs. 3 and 5 through 8. The column designations E, V, and L, refer, respectively, to the number of edges, the number of vertices, and the maximum latency of each graph. Three columns of values of T_{CP} are shown in the table. The designation Initial T_{CP} indicates the minimum clock period for each original graph; an asterisk indicates a race condition in the graph. The designation T_{CP} After Retiming indicates the minimum clock period after retiming, taking into account the clock and register delays shown in each figure. The third column of T_{CP} values was determined by retiming each graph subject to the conditions of zero clock skew and constant register delay. The value of register delay, which is different for each graph, was chosen arbitrarily as the average of all register delays in the graph.

A comparison of the values of the Initial T_{CP} with the values obtained after retiming show a dramatic improvement in minimum clock period. Furthermore, it is of interest to observe from the results obtained in Figs. 3 and 5 that the utilization of negative clock skew can achieve some improvement in maximum clock frequency compared to the zero clock skew condition (16 compared to 17 tus and 12 compared to 15 tus, respectively). On the other hand, if retiming is performed assuming zero clock skew, no local data path delay can be negative and therefore a race condition is not possible.

Table 1: Examples of Retiming using the POLY Algorithm

Example	Graph properties				T_{CP} After Retiming (tu)	T_{CP} T_{SKEW}=0 T_{REG}=const (tu)
	E	V	L	Initial T_{CP} (tu)		
Fig. 3	11	8	4	33*	16	17
Fig. 5	5	4	2	21	12	15
Fig. 6	8	6	3	28	17	15
Fig. 7	10	7	4	45*	22	16
Fig. 8	9	6	4	38*	25	16

* denotes a graph that has race conditions before retiming

VI. Conclusions

A retiming algorithm is presented which includes variable register delays and clock skew. An L matrix is introduced to calculate the clock period of a graph and to detect the existence of race conditions. In order to model the non-idealities present in actual circuits, electrical characteristic values are assigned to each edge, and these values are considered during retiming while the registers are moved from one edge to the other in the graph. In order to calculate clock skews, clock delays are assigned to each edge, where the edge represents a physical region of similar clock delay. This assignment makes possible the capability of incorporating clock distribution timing into the retiming algorithm.

Algorithm CPL is described for efficiently calculating the minimum and maximum local path delays of the L matrix. Algorithm POLY is presented which provides for retiming a graph which considers both variable register delays and clock skew. It includes merging nonlinear boolean algebra minimization with linear programming to handle these additional conditions. Because the effects of clock skew and variable register delays are an integral part of the algorithm, the process of retiming can more accurately estimate the maximum clock frequency and can possibly create a higher speed circuit.

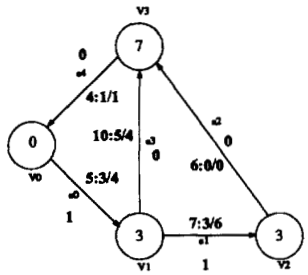

Fig. 5: Graph of synchronous circuit with latency of 2

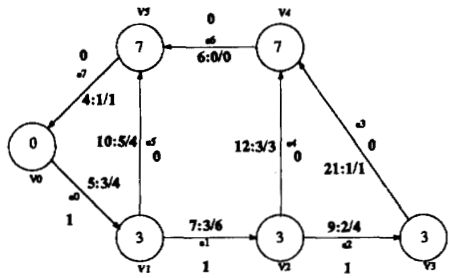

Fig. 6: Graph of synchronous circuit with latency of 3

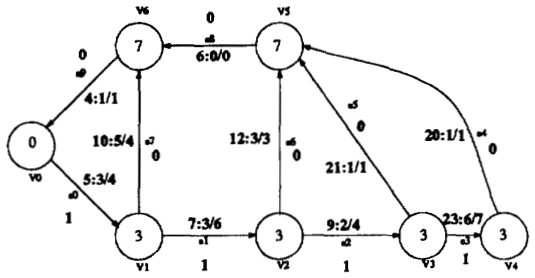

Fig. 7: Graph of three-input adder

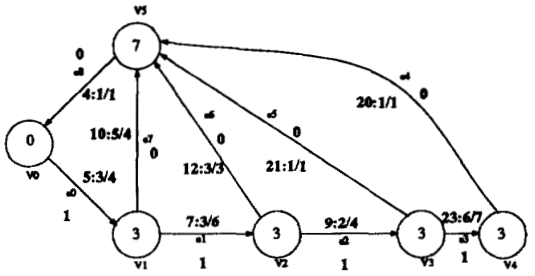

Fig. 8: Graph of four-input adder

References

[1] L. W. Cotton, "Circuit Implementation of High-Speed Pipeline Systems," *Proceedings of the Fall Joint Computer Conference*, pp. 489–504, 1965.

[2] J. R. Jump and S. R. Ahuja, "Effective Pipelining of Digital Systems," *IEEE Transactions on Computers*, vol. C-27, pp. 855–865, Sept. 1978.

[3] C. E. Leiserson and J. B. Saxe, "Optimizing Synchronous Systems," *Proceedings of 22nd Annual Symposium on Foundations of Computer Science*, pp. 23–26, Oct. 1981.

[4] P. R. Cappello and K. Steiglitz, "Bit-Level Fixed-Flow Architectures for Signal Processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-31, pp. 1016–1023, Aug. 1983.

[5] M. Hatamian and G. L. Cash, "Parallel Bit-Level Pipelined VLSI Designs for High Speed Signal Processing," *Proceedings of IEEE*, vol. 75, pp. 1192–1202, Sept. 1987.

[6] C. E. Leiserson and J. B. Saxe, "Optimizing Sunchronous Circuitry by Retiming," *Proceedings of the 3rd Caltech Conference on Very Large Scale Integration*, pp. 5–35, 1983.

[7] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, vol. 6, pp. 5–35, Jan. 1991.

[8] G. De Micheli, "Synchronous Logic Synthesis: Algorithms for Cycle-Time Minimization," *IEEE Transactions on Computer-Aided Design*, vol. CAD-10, pp. 63–73, Jan. 1991.

[9] S. Malik, E. M. Sentovich, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Retiming and Resynthesis: Optimizing Sequential Networks with Combinatorial Techniques," *IEEE Transactions on Computer-Aided Design*, vol. CAD-10, pp. 74–84, Jan. 1991.

[10] B. Lockyear and C. Ebeling, "Optimal retiming of multi-phase, level-clocked circuits," *Proceedings of the 1992 Brown/MIT Conference on Advanced Research in VLSI and Parallel Systems*, pp. 265–280, Mar. 1992.

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

[12] T. M. Burks, K. A. Sakallah, and T. N. Mudge, "Multiphase Retiming Using $minT_c$," *Proceedings of 1992 Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Mar. 1992.

[13] T. M. Burks, K. A. Sakallah, K. Bartlett, and G. Borriello, "Performance Improvement through Optimal Clocking and Retiming," *Proceedings of International Workshop on Logic Synthesis*, Feb. 1991.

[14] E. G. Friedman and J. H. Mulligan, Jr., "Pipelining of High Performance Synchronous Digital Systems," *International Journal of Electronics*, vol. 70, pp. 917–935, May 1991.

[15] E. G. Friedman and J. H. Mulligan, Jr., "Clock Frequency and Latency in Synchronous Digital Systems," *IEEE Transactions on Signal Processing*, vol. SP-39, pp. 930–934, Apr. 1991.

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

[17] M. Hatamian, *Concurrent Computations (Algorithms, Architecture and Technology)*. Understanding Clock Skew in Synchronous Systems, New York, New York:Plenum Publishing, 1988.

[18] 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, Dec. 1992.

[19] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.

[20] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, NewYork, 1976.