Monotonicity Constraints on Path Delays for Efficient Retiming with Localized Clock Skew and Variable Register Delay

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

Clock skew and delay characteristics associated with practical registers are significant factors affecting the retiming of synchronous circuits. Although work recently reported using branch and bound techniques offers a means for effective retiming taking these factors into account, the computational complexity involved is substantially greater than that associated with less general retiming algorithms that use standard linear programming methods. This paper presents sufficient conditions among values of localized clock skew and register characteristics which permit the retiming process to be achieved with a considerable reduction in computational complexity. The application of these conditions to some practical synchronous circuits is illustrated.

1 Introduction

Retiming is a technique used to increase the clock frequency of a synchronous digital pipelined system [1–8]. The locations of the registers are chosen so as to minimize the clock period while preserving the system function and latency. Although retiming can achieve a lower clock period, existing retiming algorithms do not incorporate practical circuit issues, such as variable register delay, clock skew, and interconnect delay. In an actual integrated circuit, each register receives a clock signal with a delay. The differences between the delays of the clock signals at a pair of sequentially-adjacent registers is the clock skew. These clock skews may create paths with net negative path delays, creating race conditions. By incorporating localized clock skew into retiming, a more accurate and reliable estimation of the register locations can be determined.

The authors of this paper introduced the concept of integrating these timing characteristics into the retiming process [3–5]. In that work, retiming is predicated on solving a set of nonlinear inequalities consisting of multiple choices. Although that retiming algorithm [3–5] permits the synthesis of synchronous circuits that are more accurate and reliable by including clock distribution, interconnect, and register delays, the computational requirements can be substantial when applied to large VLSI circuits.

In this paper, it is shown that by placing certain constraints on the path delays in a synchronous circuit, the multiple choice nonlinear inequalities can be eliminated, permitting the retiming problem with restricted path delays to be converted into a standard linear programming problem. Therefore, commonly used techniques for solving the linear programming problem, such as Bellman-Ford [9], can be applied to solve this set of inequalities, thereby requiring significantly less computational time. Furthermore, these restrictions on the path delays can be satisfied by changing the clock distribution network rather than the data path and in a way that minimally affects the overall system.

The paper is organized as follows. The path delay constraints are provided in Section 2. The feasibility of these constraints to practical circuits is discussed in Section 3. Experimental results comparing the run time of the constrained path delay with the unconstrained path delays are provided in Section 4. Finally, some conclusions are drawn in Section 5.

2 Path Delay Monotonicity Constraints

As described in [3–5], in order to integrate clock distribution, interconnect, and register delays into the retiming process, Register Electrical Characteristics (RECs) are attached to the edges of the graph representation of the synchronous circuit. Each REC takes the form $T_{CD} : T_{Set-up}/T_{C\rightarrow Q} - T_{Int1}/T_{Int2}$. T_{CD} is the clock delay from the global clock source to each register, T_{Set-up} is the time required for the data at the input of a register to latch, $T_{C\rightarrow Q}$ is the time required for the data to appear at the output of the register upon arrival of the clock signal, and T_{Int1} and T_{Int2} are the interconnect delay before and after the register along an edge.

The algorithm *RETSAM* [5], based on branch and bound techniques, is capable of including arbitrary register properties and arbitrary clock delays. In the event, however, that the design freedom permits the simultaneous consideration of the REC delay values and localized clock skews as part of an integrated retiming-clock distribution network design process, then, assuming certain conditions can be satisfied, the increased accuracy of the retiming process can be maintained with a substantial reduction in computational complexity. In this section, the conditions to be applied to the REC values and the localized clock skews to achieve a computationally efficient retiming are introduced.

The path delay $T_{PD}(i,j)$ of the path $e_i \rightsquigarrow e_j$ is

$$T_{PD}(i,j) = T_{C\rightarrow Q}(i) + T_{Int2}(i) + T_{Logic}(i,j) + T_{Int1}(j) + T_{Set-up}(j) + T_{Skew}(i,j), \quad (1)$$

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

where $T_{Logic}(i,j)$ is the total logic delay between edges e_i and e_j, including the interconnect delay of the zero weight edges along the path $e_i \rightsquigarrow e_j$. The clock skew T_{Skew} between two sequentially-adjacent e_i and e_j is defined as

$$T_{Skew}(i,j) = T_{CD}(i) - T_{CD}(j). \quad (2)$$

If $T_{CD}(j) > T_{CD}(i)$, the clock skew between registers i and j is defined as being negative. **Negative clock skew** occurs if the initial clock signal leads the final clock signal of a local data path. If $T_{CD}(j) < T_{CD}(i)$, the clock skew between registers i and j is positive. **Positive clock skew** occurs if the initial clock signal lags the final clock signal of a local data path. In the case that $T_{CD}(j)$ equals $T_{CD}(i)$, the clock skew is zero.

Positive clock skew increases the path delay of a local data path, potentially making its local data path a critical path, whereas negative clock skew may improve circuit speed in critical paths [10–12], however, it may also create negative path delays, resulting in **race conditions**. Race conditions are caused by clocking a register before the relevant data is successfully latched. A race condition occurs if the skew is negative and greater in magnitude than the total local data path delay [10–13]. Those paths with negative delay are called **short paths** [13]. Similarly, a **long path** designates those paths with a delay greater than the desired clock period of the circuit.

In practical integrated circuits, variations in clock delay between widely separated registers may create clock skews which can drastically affect circuit operation. An observation of (1) is that arbitrary clock skews (i.e., negative clock skews) and register delays may cause longer data paths (paths with more edges and vertices) to have less delay than shorter data paths (paths with less edges and vertices). Therefore, unless constraints are placed on the possible clock delays, the data path delays are arbitrary and can quite possibly be negative. Thus, a sub-path p_1 of a longer path p (composed of added edges and vertices) may have a delay greater than path p. An example graph in which this occurs is depicted in Figure 1. In this graph, the sub-path p_1 has a delay greater than path p. The primary cause is due to the effect of negative clock skew or to the delay of the newly placed register being smaller than the delay of the original register, effectively subtracting delay from the local data path p. The sub-path p_1 will therefore have greater delay (or the longer path p will have less delay). In the specific example shown in Figure 1, sub-path p_1 is 1 time unit (tu) greater than path p. This 1 tu difference results from the negative clock skew between edges e_1 and e_2, i.e., $T_{Skew}(e_1, e_2) = 3 - 8 = -5$ tu.

When a sub-path of a larger path has a greater delay, there are three choices for removing that path:

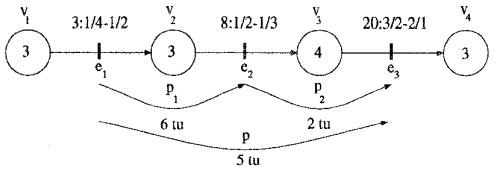

Figure 1: An example graph in which the sub-path p_1 has a delay greater than its original path p due to negative clock skew between registers.

1. Place a register between the initial and the terminal edges, since a shorter path may have a smaller delay (or a short path may have a larger delay).
2. Remove the initial edge so that the path becomes longer (more edges and vertices). This longer path may have a smaller delay.
3. Remove the terminal edge so that the path becomes longer. This longer path may have a smaller delay.

Conditions 2 and 3 are required since the data path delays are completely arbitrary and any of these conditions may possibly remove the undesirable path. Due to these three inequalities that are considered, the retiming problem with arbitrary clock and variable register delays requires solving a set of inequalities in the form of

$$x_i - x_j = a_{ij} \; or \; x_k - x_l \leq b_{ij} \; or \; x_m - x_n \leq c_{ij}. \quad (3)$$

Since these three conditions are used in *RETSAM*, standard linear programming techniques are not possible due to the boolean *or* operation [5], thereby resulting in the multiple-choice inequalities shown in (3).

A strategy to improve the time efficiency of the retiming algorithm is to place certain temporal constraints on the clock delays to guarantee that a sub-path p_1 of a larger path p will always have a larger delay, thereby removing the aforementioned conditions 2 and 3. By removing these conditions, the remaining inequalities are linear in the form of $x_i - x_j \leq a_{ij}$, since no boolean *or* operation is being performed, permitting the use of the standard Bellman-Ford method.

In Figure 2, a path with three registers and two vertices with delays $d(v_a)$ and $d(v_b)$ is depicted. For this path consisting of three registers, i, j, and k, necessary conditions for monotonically increasing path delays (smaller delays for sub-paths of larger paths) are

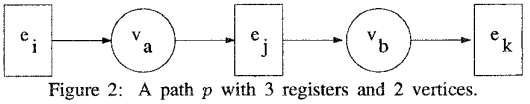

Figure 2: A path p with 3 registers and 2 vertices.

$$\begin{aligned} T_{PD}(i,k) &\geq T_{PD}(i,j), \\ T_{PD}(i,k) &\geq T_{PD}(j,k). \end{aligned} \quad (4)$$

To provide insight into how these inequalities are created, consider the graph of Figure 1. In this figure, (4) can be

used to ensure that the sub-paths p_1 and p_2 each have a delay less than the longer path p. More generally, (4) ensures that paths $e_i \rightsquigarrow e_j$ and $e_j \rightsquigarrow e_k$ each have a delay less than the longer path $e_i \rightsquigarrow e_k$.

Defining $T_{REG}(a)$ and $T_{REG}(b)$ as the total set-up and clock-to-Q delays of edges a and b, respectively,

$$\begin{aligned} T_{REG}(a) &= T_{Set-up}(a) + T_{C\rightarrow Q}(a), \\ T_{REG}(b) &= T_{Set-up}(b) + T_{C\rightarrow Q}(b), \end{aligned} \tag{5}$$

the following inequalities are obtained from (1) and (5).

$$\begin{aligned} T_{PD}(j,k) &\geq T_{REG}(j), \\ T_{PD}(i,j) &\geq T_{REG}(j). \end{aligned} \tag{6}$$

Assume a path with two edges a and b, and a vertex between these edges. The following condition is required to ensure monotonically increasing path delays with increasing path length,

$$T_{PD}(a,b) \geq \max\{T_{REG}(a), T_{REG}(b)\}. \tag{7}$$

Equation (7) sets a lower limit on the delay of a local data path to ensure that only one inequality is required. There is also an upper limit that can be defined as

$$T_{PD}(a,b) \leq C, \tag{8}$$

where C is the maximum permitted clock period of the circuit. This upper limit is used to guarantee that each local data path has a delay smaller than the maximum permitted clock period of the synchronous circuit.

Since $T_{PD}(a,b)$ depends on the clock delays driving the initial and final registers of the local data path between edges e_a and e_b, (7) is a constraint which is imposed on the clock distribution network. Equation (7) can be expanded into (9), which describes the specific constraint on the individual clock delays.

$$T_{CD}(a) - T_{CD}(b) \geq -\tau(a,b), \tag{9}$$

$\tau(a,b)$ denotes a constant depending only on the REC parameters attached to edges a and b. $\tau(a,b)$ can be calculated from (10) and is the "**negative clock skew tolerance of the local data path from edge a to edge b.**"

$$\begin{aligned} \tau(a,b) = T_{C\rightarrow Q}(a) + T_{Int2}(a) + d(v_a) + \\ T_{Set-up}(b) + T_{Int1}(b) - \max\{T_{REG}(a), T_{REG}(b)\}. \end{aligned} \tag{10}$$

The clock skew of any local data path cannot be more negative than the clock skew tolerance of that local data path. An observation of (7) is that strictly positive local data path delays are sufficient for computationally inexpensive retiming. For computational efficiency, retiming is best performed on a properly designed clock distribution network in which the negative clock skew of a local data path does not exceed the maximum tolerance of that path, as determined by (10). This strategy therefore does not verify the existence of race conditions but instead assumes that all race conditions have been eliminated *a priori.*

If (9) is satisfied, a clock distribution network which permits computationally efficient retiming is possible. Equations (8) and (9) can be rewritten as

$$\begin{aligned} T_{CD}(b) - T_{CD}(a) &\leq \tau(a,b), \\ T_{CD}(a) - T_{CD}(b) &\leq \mathrm{T}(a,b). \end{aligned} \tag{11}$$

where $\mathrm{T}(a,b)$ is the "**positive clock skew tolerance of the local data path** $e_i \rightsquigarrow e_j$" and can be calculated as

$$\begin{aligned} \mathrm{T}(a,b) = C - T_{C\rightarrow Q}(a) - T_{Int2}(a) - \\ d(v_a) - T_{Set-up}(b) - T_{Int1}(b). \end{aligned} \tag{12}$$

3 Example of Monotonicity Constraints

An example of the application of these conditions on the clock delays to a practical circuit is considered in this section. The digital correlator presented by Leiserson-Saxe [1] is used as an example circuit. The logic elements on the vertices v_1, v_2, v_3, and v_4 are comparators and are modeled as XNOR gates with a nominal delay value of 3.5 ns. The logic elements on vertices v_5, v_6, and v_7 are full adders with a nominal delay value of 4.0 ns. Typical register set-up and clock-to-Q times of 4.0 ns and 3.0 ns are used. Interconnect delays, T_{Int1} and T_{Int2}, are assumed to each be 1.4 ns, approximately 20% of the register delays. Zero clock skew is initially assumed to predict the negative clock skew tolerance of a path.

For edges e_0 and e_1, the register delays are

$$\begin{aligned} T_{REG}(e_0) = T_{REG}(e_1) = 4.0 + 3.0 = 7.0 \text{ ns}, \\ \Rightarrow \max\{T_{REG}(e_0), T_{REG}(e_1)\} = 7.0 \text{ ns}. \end{aligned}$$

Assuming $T_{Skew}(e_0, e_1) = 0$, the path delays are

$$\begin{aligned} T_{PD}(e_0,e_1) = T_{C\rightarrow Q}(e_0) + T_{Int2}(e_0) + d(v_1) + T_{Int1}(e_1) + T_{Set-up}(e_1) \\ + T_{Skew}(e_0,e_1) = 3 + 1.4 + 3.5 + 1.4 + 4 + T_{Skew}(e_0,e_1) = 13.3 \text{ ns}. \end{aligned}$$

From (7), $T_{PD}(e_0,e_1) \geq \max\{T_{REG}(e_0), T_{REG}(e_1)\}$, which is satisfied, since 13.3 ns $\geq$ 7.0 ns. These conditions are satisfied for all the other paths in the digital correlator. Therefore, as long as the clock skew is zero, (7) is satisfied, and the digital correlator has monotonically increasing path delays throughout the entire circuit. Now, let $T_{Skew}(e_0,e_1) = T_{CD}(e_0) - T_{CD}(e_1)$ in the correlator be negative. Positive clock skew is not considered here since

it does not affect the monotonicity constraint. Applying negative clock skew, the inequalities become

$$T_{Skew}(e_0, e_1) + 13.3 \text{ ns} \geq 7 \text{ ns} \Rightarrow T_{Skew} \geq -6.3 \text{ ns}.$$

Thus, the negative clock skew tolerance of the local data path is 6.3 ns. A methodology for designing clock distribution networks based on non-zero localized clock skew is described in [14]. Algorithms and circuit delay models are provided for implementing the topology and circuit elements within a clock distribution network such that the specific monotonicity constraints can be satisfied.

4 Comparison of Computational Complexity

Diverse strategies have been used to design clock distribution networks which in turn effect the localized clock skews. To accomodate arbitrarily selected values of localized clock skew in the retiming process with real register characteristics, retiming approaches, such as *RETSAM* [5], seem indicated. The foregoing results, however, indicate that the use of the monotonicity constraints in (11) enables the retiming process to use standard linear programming techniques (*OPT2* in [1]) which require much less computational time. As an illustration of some differences in the computation times using the two approaches for specifying the clock delays, several retiming computations were made. The structures of the synchronous circuits were taken from the MCNC benchmark circuits [15] with the same REC characteristics and are compared in Table 1.

The initial four columns describe the properties of the modified benchmark circuits. These properties are 1) the name of the benchmark example as it appears in the MCNC archive, 2) the number of edges and 3) vertices in the graph of each circuit, and 4) the latency of the circuit. The fifth and sixth columns contain the minimum clock period of the retimed circuit with and without restricted path delays, respectively. As is shown in Table 1, by using standard linear programming methods rather than branch and bound algorithms, the run time of the retiming process is significantly enhanced.

Table 1: Comparison of the CPU times of the retiming algorithm with constrained and unconstrained path delays

Example	Graph properties			CPU time using *OPT2* [1] (sec)	CPU time using *RETSAM* [5] (sec)
	Edges	Vertices	Latency		
LGSynth89 - multi-level (netblif)					
C17	26	19	6	1.423	963.1
cm82a	59	37	4	1.919	1164
majority	26	17	5	1.142	507.3
LGSynth91 - multi level (blif)					
cm85a	70	36	3	1.953	3306
cm151a	38	22	3	1.447	795.2

5 Conclusions

Path delay monotonicity constraints are presented in this paper to permit the use of standard linear programming methods for solving the retiming problem with electrical delay information, such as localized clock skew and variable register delay. The feasibility of these constraints to practical circuits is discussed. Noteworthy is the degree to which the design of the clock distribution network and the retiming process are inextricably intertwined.

A branch and bound based algorithm for solving the general retiming problem which considers electrical information has been demonstrated on MCNC benchmark circuits and compared with standard linear programming methods to solve the retiming problem for circuits with constrained path delays. It is observed that by constraining the path delays, run times are significantly improved by up to three orders of magnitude for the tested circuits belonging to the MCNC benchmark set. The improvement in computational efficiency increases with increasing circuit size.

References

[1] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, Vol. 6, pp. 5–35, January 1991.

[2] G. De Micheli, "Synchronous Logic Synthesis: Algorithms for Cycle-Time Minimization," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-10, No. 1, pp. 63–73, January 1991.

[3] T. Soyata, E. G. Friedman, and J. H. Mulligan, Jr., "Integration of Clock Skew and Register Delays into a Retiming Algorithm," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1483–1486, May 1993.

[4] T. Soyata and E. G. Friedman, "Synchronous Performance and Reliability Improvement in Pipelined ASICs," *Proceedings of the ASIC Conference*, pp. 383–390, September 1994.

[5] T. Soyata and E. G. Friedman, "Retiming with Non-Zero Clock Skew, Variable Register, and Interconnect Delay," *Proceedings of the IEEE International Conference on Computer-Aided Design*, November 1994.

[6] B. Lockyear and C. Ebeling, "The Practical Application of Retiming to the Design of High-Performance Systems," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 288–295, November 1993.

[7] B. Lockyear and C. Ebeling, "Optimal Retiming of Level-Clocked Circuits Using Symmetric Clock Schedules," *IEEE Transactions on Computer-Aided Design*, Vol. 13, No. 9, pp. 1097–1109, September 1994.

[8] S. Simon, E. Bernard, M. Sauer, and J. A. Nossek, "A New Retiming Algorithm for Circuit Design," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 4.35–4.38, May/June 1994.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw-Hill, 1990.

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

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

[12] E. G. Friedman, "Clock Distribution Design in VLSI Circuits — an Overview," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1475–1478, May 1993.

[13] K. A. Sakallah, T. N. Mudge, T. M. Burks, and E. S. Davidson, "Synchronization of Pipelines," *IEEE Transactions on Computer-Aided Design*, Vol. CAD–12, No. 8, pp. 1132–1146, August 1993.

[14] J. L. Neves and E. G. Friedman, "Synthesizing Distributed Buffer Clock Trees for High Performance ASICs," *Proceedings of the IEEE ASIC Conference*, pp. 126–129, September 1994.

[15] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0," Tech. Rep., Microelectronics Center of North Carolina, January 1991.