DEMONSTRATION OF SPEED ENHANCEMENTS ON AN INDUSTRIAL CIRCUIT THROUGH APPLICATION OF NON-ZERO CLOCK SKEW SCHEDULING

Dimitrios Velenis[1], *Kevin T. Tang*[2*], *Ivan S. Kourtev*[3], *Victor Adler*[4*], *Franklin Baez*[5], *and Eby G. Friedman*[1]

[1]Department of Electrical and Computer Engineering, University of Rochester, Rochester, New York 14627
[2]Broadcom Corporation, 2099 Gateway Place, San Jose, California 95110
[3]Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania 15261
[4]Sun Microsystems, 901 San Antonio Road, Palo Alto, California 94303
[5]Intel Corporation, 2200 Mision College Boulevard, Santa Clara, California 95052

ABSTRACT

A demonstration of the application of non-zero clock skew scheduling to enhance the speed characteristics of several functional unit blocks in a high performance processor is presented. It is shown that non-zero clock skew scheduling can improve circuit performance while relaxing the strict timing constraints of the critical data paths within a high speed system. A software tool implementing a non-zero clock skew scheduling algorithm is described together with a methodology that generates the required clock signal delays by replacing clock buffers from predesigned cell libraries. Timing margin improvements of up to 18% are achieved through the application of non-zero clock skew scheduling in certain functional blocks of an industrial high performance microprocessor.

1. INTRODUCTION

Most high performance digital integrated circuits utilize fully synchronous timing, requiring a reference signal to control the temporal sequence of operations. This synchronous time reference is provided by a globally distributed signal, typically called the clock signal. Due to the vital role of the clock signal in the operation of a synchronous system, the clock signal and the related clock distribution network require careful planning and design. As the on-chip feature sizes are reduced concurrently with increasing chip dimensions, the on-chip interconnect impedances have become increasingly significant, of greater importance than the active device delay. Due to the interconnect impedances, the delay of the clock signal arriving at different locations within a circuit may vary significantly, possibly causing synchronization failures. Enhanced design approaches are therefore required for efficiently implementing the clock distribution network in order to prevent any deleterious effects within the circuit.

Many of the techniques that have been developed to improve the performance and design efficiency of a clock distribution network target minimal (or zero) clock skew between each pair of sequentially-adjacent registers [1]. This design methodology is called *zero clock skew scheduling* and is implemented in many different ways such as inserting distributed buffers within the clock tree [2], using symmetric distribution networks, such as H-tree structures [3] to minimize the clock skew, and applying zero skew clock routing algorithms [4, 5] to automatically layout high speed clock distribution networks.

Minimum (or zero) clock skew scheduling has been used in many high performance circuits. Intel Corporation applies a minimum clock skew methodology with localized tuning in the design of their latest microprocessors, including the Itanium^{TM}†, the first processor in the Intel's IA-64 microarchitecture family [6, 7]. Simulations on a high performance processor demonstrate that a significant improvement in circuit performance can be achieved while minimizing the likelihood of race conditions with the application of a non-zero clock skew schedule.

The effectiveness of the application of non-zero clock skew scheduling [1, 8] to a high performance industrial microprocessor is demonstrated in this paper. It is shown that significant improvements in circuit speed can be achieved with clock skew scheduling. The paper is organized as follows. Background information on clock skew and data path timing constraints and hazards is summarized in Section 2. An algorithm that specifies the optimal clock skew schedule for a circuit and a related software implementation is described in Section 3. In Section 4, simulation results illustrating the speed improvements are described. Finally, some conclusions are presented in Section 5.

This research was supported by a grant from Intel Corporation.

*Drs. Kevin T. Tang and Victor Adler contributed to the development of this project during summer internships at Intel Corporation, prior to receiving their Ph.D. degree from the University of Rochester.

†Itanium^{TM} is a registered trademark of Intel Corporation

0-7803-7057-0/01/$10.00 ©2001 IEEE.

2. BACKGROUND

A digital synchronous circuit is composed of a network of functional logic elements and globally clocked registers. Two registers, R_i and R_j, in a synchronous digital circuit are considered sequentially-adjacent if there exists at least one sequence of logic elements and/or interconnect connecting the output of the initial register R_i to the input of the final register R_j. A pair of sequentially-adjacent registers together with a logic block and/or interconnect make up a local data path. A data path consisting of one or more local data paths is called a global data path. A local data path composed of two registers, R_i and R_j, driven by the clock signals, C_i and C_j respectively, is shown in Fig. 1.

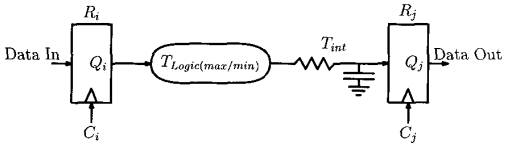
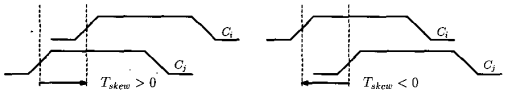


Figure 1: A local data path.

The difference in clock signal arrival times between two sequentially-adjacent registers is called the *local clock skew* [1]. More specifically, given two sequentially-adjacent registers, R_i and R_j, the clock skew between these two registers is defined as $T_{skew} = T_{CD_i} - T_{CD_j}$, where T_{CD_i} and T_{CD_j} are the clock delays from the clock source to the registers, R_i and R_j , respectively. If the clock delay to the initial register T_{CD_i} is greater than the clock delay to the final register T_{CD_j}, the clock skew is described as positive. Similarly, if the clock delay to the initial register T_{CD_i} is less than the clock delay to the final register T_{CD_j}, the clock skew is described as negative. Waveforms exemplifying positive and negative clock skew for the local data path shown in Fig. 1 are illustrated in Fig. 2 [1].



(a) Positive clock skew (b) Negative clock skew

Figure 2: Examples of positive and negative clock skew.

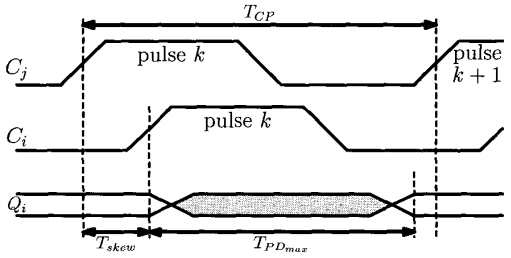
The strategy of minimizing clock skew has been a central design technique for decades in digital synchronous circuit design methodologies. Zero (or minimal) clock skew methods require the clock delay from the clock source to each register of the system to be approximately equal. As described by Fishburn in [8], further optimization of the circuit performance and reliability can be achieved by applying non-zero clock skew in some (or all) of the local data paths. The individual clock skew for each local data path is determined by satisfying

Table 1: Timing relationships for a local data path R_i to R_j

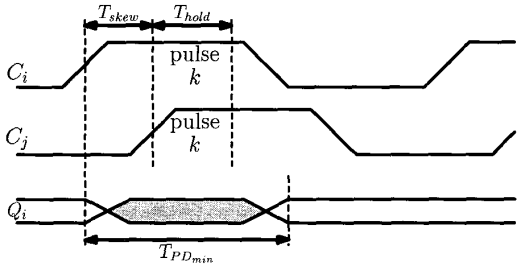
$T_{CP} \geq T_{skew} + T_{PD_{max}}$	(1)
$T_{PD_{min}} \geq T_{skew} + T_{hold}$	(2)
$T_{PD_{max}} = T_{C-Q_i} + T_{Logic(max)} + T_{int} + T_{set-up}$	(3)
$T_{PD_{min}} = T_{C-Q_i} + T_{Logic(min)} + T_{int} + T_{set-up}$	(4)

specific timing relationships and conditions in order to minimize the system-wide clock period while avoiding all race conditions. For the local data path from register R_i to register R_j, shown in Fig. 1, these timing relationships are listed in Table 1.

In the inequalities listed in Table 1, T_{skew} is the clock skew between registers R_i and R_j. $T_{PD_{max}}$ ($T_{PD_{min}}$) is the maximum (minimum) propagation delay between registers, R_i and R_j, shown in (3) and (4), respectively. $T_{Logic(max)}$ ($T_{Logic(min)}$) is the maximum (minimum) propagation delay of the logic block between the registers R_i and R_j. T_{hold} is the time that the input data signal must be stable at register R_j once the clock signal changes state. T_{set-up} is the time required for the data to successfully propagate to and be latched within the register R_j. T_{C-Q_i} is the time required for the data signal to leave R_i once the register is enabled by the clock pulse C_i. T_{int} represents the temporal effect of the interconnect impedance on the path delay between the registers, R_i and R_j [9, 10]. T_{CP} is the minimum clock period.



(a) To prevent zero clocking, $T_{CP} \geq T_{skew} + T_{PD_{max}}$



(b) To prevent double clocking, $T_{PD_{min}} \geq T_{skew} + T_{hold}$

Figure 3: Timing hazards in synchronous digital systems.

From the inequalities listed in Table 1, (1) guarantees that the data signal released from R_i is latched into R_j before the next clock pulse arrives at R_j, preventing *zero clocking* [8]. Also, (2) prevents latching an incorrect data signal into R_j by the clock pulse that latched the same data signal into R_i, or *double clocking* [8]. This race condition is created when the clock skew is negative and greater in magnitude than the path delay. If the clock skew is negative but smaller than the path delay, this effect can be used to improve circuit performance. This method of improving performance is called clock skew scheduling [1, 8, 12, 13]. Timing relationships that prevent double and zero clocking are shown in Figs. 3(a) and 3(b), respectively.

For a given clock period T_{CP}, (1) and (2) determine a range within which each local clock skew T_{skew} can vary. This tolerance range is described here as the *permissible clock skew range* [10, 11] between the minimum permissible clock skew $T_{skew(min)}$ and the maximum permissible clock skew $T_{skew(max)}$. The permissible clock skew range varies for different data paths since $T_{PD_{min}}$ and $T_{PD_{max}}$ depend on the delay characteristics of each local data path. $T_{skew(max)}$ is zero for those critical local data paths that limit the minimum clock period T_{CP} of the entire system.

The inequalities (1) and (2) listed in Table 1 are sufficient conditions to determine an optimal clock skew schedule, the associated minimum clock path delays, and the allowed variation of the clock skew for each local data path. In this way, the minimum clock period is determined such that the overall circuit performance is maximized while eliminating any race conditions.

3. ALGORITHM IMPLEMENTATION

The optimal clock scheduling problem has been described in [8] as a set of linear inequalities which can be solved with standard linear programming techniques. An algorithm for determining the minimum clock period based on the overlapping of permissible ranges of the clock skew between different data paths has been described in [10, 11]. These concepts have been further enhanced, implemented as an algorithm, integrated into a software tool [12–14], and applied to a functional unit within a high performance microprocessor to determine an optimal clock skew schedule. The results of this evaluation are the primary new results presented in this paper.

The development of a software tool to implement an optimum clock scheduling algorithm is described in [10–13]. The input data to this tool are the minimum and maximum delays of each of the local data paths of the circuit (as well as the initial clock delays to each of the registers). With this information, the software tool specifies an optimal clock skew schedule for the circuit; specifically, the minimum clock period that maximizes circuit performance and the associated clock path delays from the clock source to the individual registers that satisfy the target clock skew schedule. The steps of the implemented algorithm are as follows:

1. A graph model of the circuit is produced that describes the input circuit C. Each vertex of the graph represents a register within C. Each arch of the graph connecting two vertices represents a local data path in C.

2. The current clock period for the circuit C is determined. The current clock period is the arithmetic mean of two bounding values. The upper bound is initially set equal to the maximum delay of all of the data paths belonging to C. The lower bound is initially set equal to the greatest difference between the maximum and minimum propagation delay of each local data path within C.

3. Using the clock period specified from step 2, the permissible clock skew range is calculated from (1) and (2) for each pair of sequentially-adjacent registers in C.

4. The permissible range of the clock skew of the global data paths is specified by the intersections of the permissible ranges of the local data paths calculated in the previous step. If the intersection is empty, no feasible clock schedule exists for the clock period specified in step 2.

5. If a feasible clock schedule results from step 4, the algorithm iterates to step 2, and the current clock period specified in the previous iteration becomes the upper bound and is marked as a possible optimum solution. If a non-feasible clock schedule results from step 4, the algorithm iterates again to step 2 and the previously specified current clock period becomes the lower bound.

Iterations of the algorithm between steps 2 and 5 continue until the difference between the upper and lower bounds of the clock period is less than a specified positive number ε. The last clock period marked as a possible optimum solution is the minimum achievable clock period for the circuit C. Based on this clock period, (1), and (2), the clock skew between each pair of sequentially-adjacent registers within C is computed.

6. The final step of the algorithm assigns the clock path delay to each of the registers within C. For each global data path, the individual clock delays from the clock source to the registers are calculated by first assigning the delay to the registers of the local data path with the largest clock skew value. The delays to the other registers are assigned by using the relative clock skew values among the remaining registers within the global data path.

The optimality of the solution depends solely upon the value of the constant ε that controls the number of approximating iterations executed by the algorithm. Reducing the value of ε reduces the distance between the minimum clock period determined by the algorithm and the minimum clock period set by (1) and (2). The choise of ε is a tradeoff between performance and computational run time of the algorithm.

4. EXPERIMENTAL RESULTS FROM THE APPLICATION OF OPTIMUM CLOCK SKEW SCHEDULING

In a joint research project between the University of Rochester and Intel Corporation, the process of enhancing the speed and power dissipation [15] of an industrial circuit through the application of non-zero clock skew scheduling has been investigated. Specifically, the application of clock skew scheduling to certain (highly tuned) functional blocks within a high performance microprocessor has been evaluated. It is shown here that the application of non-zero clock skew scheduling to these circuits yields an improvement in timing margins of up to 18% within the data paths of certain functional unit blocks (FUBs).

The clock scheduling tool described in Section 3 [12, 13] has been applied to specific FUBs within a high performance microprocessor. A circuit diagram of one of these FUBs is shown in Fig. 4 with normalized maximum and minimum local data path delays. All of the timing information in the following analysis is described in terms of these normalized path delays.

The initial clock period for the FUB shown in Fig. 4 is 35 tu (time units). By exploiting the differences in the maximum delays between data path A and the three parallel data paths, B, C, and D, the clock period can be reduced from 35 tu to 28 tu. This 20% performance improvement can be achieved through application of a negative clock skew of -7 tu to data path A by adding 7 tu to the clock path delay from the clock source to register R_2. In this case, the time available for the data signal to propagate along data path A is $T_{CP} + T_{skew}$ = 28 + 7 = 35 tu. The time available for a data signal to propagate along the longest of the data paths between registers R_2 and R_3 (data path B) is 28 - 7 = 21 tu. Note that data paths F and G can also be synchronized by a clock period of 28 tu without violating any timing constraints. Thus, an approximately 20% improvement in circuit performance can be achieved by applying a non-zero clock skew schedule to this specific FUB.

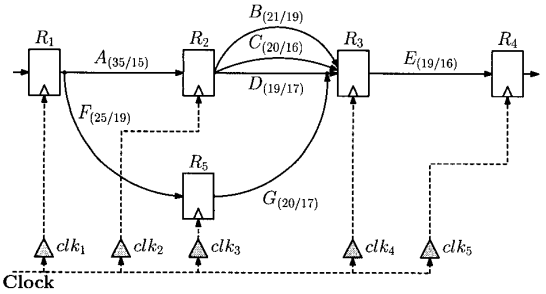


Figure 4: Circuit graph of Itanium^{TM} FUB with normalized data path delays.

The added delay to the path from the clock source to register R_2 is accomplished by decreasing the size of the clock buffer (clk_2 shown in Fig. 4) that sources the clock signal that drives this register. This delay change is accomplished by replacing the clock buffer with another slower buffer from a predesigned cell library. In this way, the clock signal delay can be increased without requiring the redesign of the original clock buffer. Several different sizes of predesigned clock buffers that drive register R_2 have been evaluated. The variation of the clock signal delay to different clock buffer sizes is shown in Fig. 5

As illustrated in Fig. 5, the clock delay from the clock source to a register is inversely proportional to the size of the clock buffer. This behavior is due to the increased output resistance of the smaller sized buffers, resulting in reduced current flow which introduces additional delay to the clock signal [16]. The clk_2 buffer that is initially used in the specific FUB (see Fig 4) is buffer No. 6 with a delay of 9.67 tu (see Fig 5). In order to produce an additional clock delay of 7 tu to drive register R_2, buffer No. 4 is used. The additional signal delay is $16.27 - 9.67 = 6.60$ tu, only a 5.7% error from the target value of 7 tu. The minimum clock signal period that is achieved with this clock skew schedule is an 28.4 tu, producing an 18.8% improvement in speed.

Decreasing the size of the clock buffer in order to increase the delay of the clock line has an additional ben-

Buffer Number	Normalized buffer size	Normalized clock signal delay (tu)
1	1.00	24.93
2	1.43	20.14
3	1.71	17.79
4	2.05	16.27
5	6.07	10.90
6	10.47	9.67

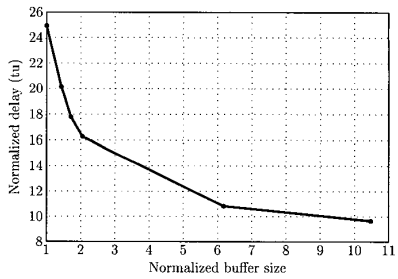


Figure 5: Variation of clock signal delay to different clock buffer sizes.

eficial effect on the power dissipation, since the current flowing through the buffer is reduced. For the target circuit that contains the slower clock buffers, the power saving is approximately 1% of the total power consumed by this block.

5. CONCLUSIONS

Simulations of specific FUBs within a high performance microprocessor demonstrate that improvements in the timing margin of the data paths can be achieved by applying non-zero clock skew. It is shown that in specific circuit blocks the timing margin can be increased by up to 18% by exploiting the differences in propagation delays between sequentially-adjacent data paths. The required clock delays to the individual registers can be achieved by replacing the clock buffers that drive these registers with buffer cells from a predesigned cell library. A non-zero clock skew scheduling software tool has also been developed. This tool has been evaluated on a number of industrial circuits, demonstrating the general utility of clock skew scheduling to improve the timing characteristics of a synchronous digital system.

6. REFERENCES

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

[2] E. G. Friedman and S. Powell, "Design and Analysis for a Hierarchical Clock Distribution System for Synchronous Standard Cell/macrocell VLSI," *IEEE Journal of Solid-State Circuits*, Vol. SC-21, No. 2, pp. 240-246, April 1986.

[3] H. B. Bakoglou, J. T. Walker, and J. D. Meindl, "A Symmetric Clock-Distribution Tree and Optimized High-Speed Interconnections for Reduced Clock Skew in ULSI and WSI Circuits," *Proceedings of the IEEE International Conference on Computer Design*, pp. 118-122, October 1986.

[4] T.- H. Chao, Y.-C. Hsu, J. -M. Ho, K. D. Boese, and A. B. Kahng. "Zero Skew Clock Routing with Minimum Wirelength," *IEEE Transactions Circuits Systems II: Analog and Digital Signal Processing*, Vol. 39, No. 11, pp. 799-814, November 1992.

[5] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Boston, Massachusetts: Kluwer Academic Publishers, 1995.

[6] U. Desai, S. Tam, R. Kim and J. Zhang, "Itanium Processor Clock Design," *Proceedings of the ACM/SIGDA International Symposium on Physical Design*, pp. 94-98, April 2000.

[7] S. Rusu and S. Tam, "Clock Generation and Distribution for the First IA-64 Microprocessor," *Proceedings of the IEEE International Solid State Circuits Conference*, pp. 176-177, February 2000.

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

[9] J. L. Neves and E. G. Friedman, "Design Methodology for Synthesizing Clock Distribution Networks Exploiting Non-Zero Clock Skew," *IEEE Transactions on VLSI Systems*, Vol. VLSI-4, No. 2, pp. 286-291, June 1996.

[10] J. L. Neves and E. G. Friedman, "Buffered Clock Tree Synthesis with Non- Zero Clock Skew Scheduling for Increased Tolerance to Process Parameter Variations," *Journal of VLSI Signal Processing*, Volume 16, Numbers 2/3, pp. 149- 161, June/July 1997.

[11] 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.

[12] I. S. Kourtev and E. G. Friedman, *Timing Optimization Through Clock Skew Scheduling*, Norwell, Massachusetts: Kluwer Academic Publishers, 2000.

[13] I. S. Kourtev and E. G. Friedman, "Synthesis of Clock Tree Topologies to Implement Non-Zero Skew Schedule," *IEE Proceedings-Circuits, Devices and Systems*, Volume 146, No. 6, pp. 321-326, December 1999.

[14] I. S. Kourtev and E. G. Friedman, "Clock Skew Scheduling for Improved Reliability via Quadratic Programming," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 239-243, November 1999.

[15] D. Velenis, K. T. Tang, I. S. Kourtev, V. Adler, F. Baez, and E. G. Friedman, "Demonstration of Speed and Power Enhancements through Application of Non-Zero Clock Skew Scheduling," *Proceedings of the ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pp. 58-63, December 2000.

[16] V. Adler and E. G. Friedman, "Repeater Design to Reduce Delay and Power in Resistive Interconnect," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. CAS II-45, No 5, pp. 607-616, May 1998.