Demonstration of Speed and Power Enhancements through Application of Non-Zero Clock Skew Scheduling

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

[1]Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627
[2] Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15261
[3] Sun Microsystems, 901 San Antonio Road, Palo Alto, CA 94303
[4] Intel Corporation, 2200 Mission College Blvd, Santa Clara, CA 95052

ABSTRACT

A demonstration of the application of non-zero clock skew scheduling to enhance the speed and power 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. Furthermore, methods for saving power by slowing down the non-critical data paths are also described. Timing margin improvements of up to 20% and decreased power consumption of up to 80% are achieved in certain functional blocks of a 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 skew clock 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 microprocessor is demonstrated in this paper. It is shown that significant improvements in both circuit speed and power dissipation 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. A strategy for minimizing power by delaying the non-critical data paths and exploiting non-zero clock skew is discussed in Section 3 . In Section 4, an algorithm that specifies the optimal clock skew schedule and simulation results illustrating the speed and power improvements are described. Finally, conclusions are presented in Section 5.

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 digital synchronous circuit

*Dr. Victor Adler contributed to the development of this project during an internship at Intel in the summer of 1997, prior to receiving his Ph.D. degree from the University of Rochester in late 1998.

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

TAU'00, December 4-5, 2000, Austin, Texas.

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.

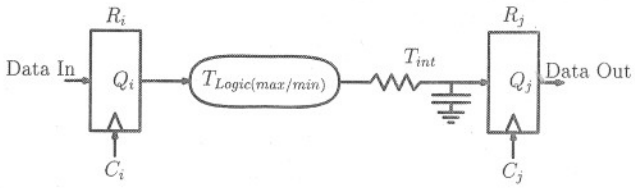
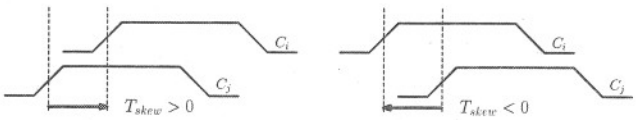


Figure 1: A local data path.

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

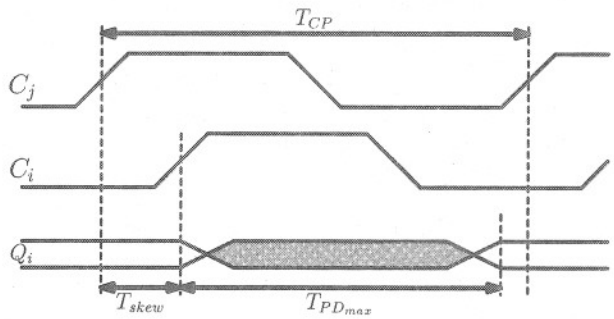
Figure 2: Examples of positive and negative clock skew.

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 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

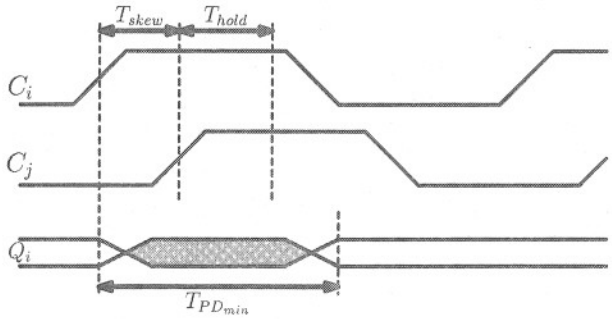
Table 1: Timing relationships and definitions for a local data path R_i to R_j

$T_{CP} \geq T_{skew} + T_{PD_{max}}$	(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)

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 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 to leave R_i once the register is enabled by the clock pulse C_i. T_{int} is 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) Preventing zero clocking: $T_{CP} \geq T_{skew} + T_{PD_{max}}$



(b) Preventing double clocking: $T_{PD_{min}} \geq T_{skew} + T_{hold}$

Figure 3: Prevention of timing hazards.

From the inequalities listed in Table 1, (1) guarantees that the data signal latched in R_i is latched into R_j before the next clock pulse arrives in 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 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 each local data path. $T_{skew(max)}$ is zero for those critical local data paths that determine the minimum clock period T_{CP} of the entire system.

The inequalities listed in Table 1 are sufficient conditions to determine an optimal clock skew schedule, the associated minimum clock path delays, the allowed variation of the clock skew for each local data path, and the minimum clock period such that the overall circuit performance is maximized while eliminating any race conditions. The optimal clock scheduling problem has been described as a set of linear inequalities which can be solved with standard linear programming techniques [8]. 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, 13], and applied to a functional unit within a high performance microprocessor to determine the optimal clock skew schedule. A detailed description of the application of this tool together with simulation results are presented in Section 4.

3. POWER SAVINGS IN NON-CRITICAL DATA PATHS

Modern digital synchronous integrated circuits consist of millions of logic gates. The number of local data paths in these systems is therefore large. The clock frequency of these systems is constrained by those local data paths with the longest delay. These data paths are called the critical (or worst case) paths. Good control of the timing of these paths is required. As described in [1, 8, 12], application of non-zero clock skew scheduling can result in increased system performance by relaxing the strict timing constraints of these critical data paths. Small amounts of localized clock skew are often used to enhance the timing margins of the critical data paths, even in those high performance circuits that are designed, in general, as "minimum clock skew" circuits such as the Itanium^{TM} microprocessor [6, 7].

In a large high performance system, such as a microprocessor, the number of critical data paths is small as compared with the total number of data paths in the system. For example, in a specific system described in [12], less than 5% of the total data paths are within 20% of the maximum path delays while more than 65% of the total data paths have path delays less than half of the maximum path delay.

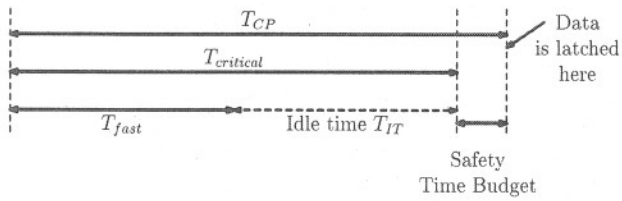
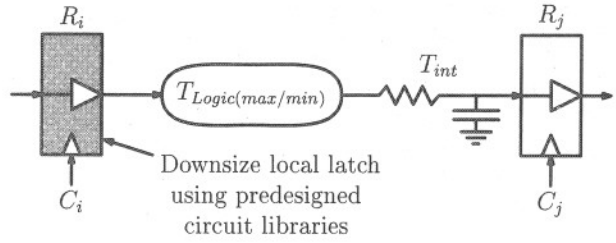


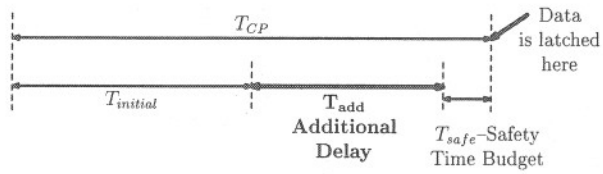
Figure 4: Short data path delay as compared to a critical long data path delay.

Alternatively, more than 65% of the local data paths are at least twice as fast as compared with the slowest local data paths. A similar distribution of path delays is common in the majority of high complexity circuits.

The short data paths of a system are synchronized by the same clock signal that synchronizes the critical long data paths. Therefore, idle time (T_{IT}) exists in these short data paths since the data signal arrives at the final register well before the clock signal arrives at the same register, as shown in Fig. 4. This idle time can be exploited to slow down these short data paths in order to save power. One way to accomplish this technique is by downsizing the latch R_i that drives the data path as shown in Fig. 5(a). By down-



(a) A data path with a downsized latch to decrease the power of the fast data paths



(b) The added delay of the fast data path does not violate the long path timing constraint

Figure 5: Increasing the delay of the fast data paths by downsizing the local latches that drive these paths.

sizing the latch, the geometric width of the output driver within the latch is decreased and the output current of the latch is therefore reduced, resulting in a decrease in power consumption, albeit with an increase in the data path delay.

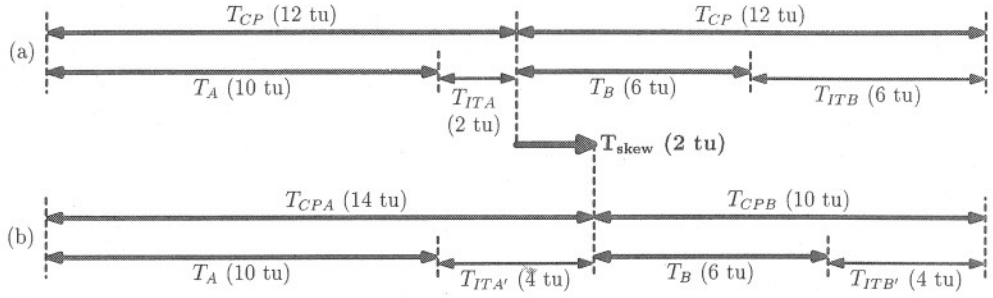


Figure 6: Application of local clock skew to equalize the available idle time between the long and short delay data paths. (a) Initial timing of the data paths. (b) Timing of the data paths after the application of local clock skew

There are constraints, however, that limit the minimum size of an output driver and thereby the additional delay that is introduced. One is that the additional delay should not violate the maximum path delay constraint as shown in Fig. 5(b). The summation of the initial data path delay $T_{initial}$, the additional delay T_{add}, and the safety time budget T_{safe} should be less (or in the worst case equal) to the clock period T_{CP}. Another constraint is that the introduction of smaller sized output drivers affects the rise and fall times of the data path signal. Therefore, the effect of the total gate capacitance and the parasitic interconnect capacitance should be considered.

This concept of slowing down the fast data paths in order to save power can be further applied to slower, more critical data paths with the aid of non-zero clock skew scheduling. By applying negative clock skew to these slower data paths, the idle time in these data paths can be expanded, permitting these paths to be further slowed down. However, there is one condition that must be satisfied for this concept to be feasible. This condition is that the data path that follows the slow data path should be sufficiently fast to eliminate a zero clocking hazard as described by (1). Application of this concept to long data path delays is shown in Fig. 6. As shown in Fig. 6(a), data path A has a long delay of T_A = 10 tu and data path B has a short delay of T_B = 6 tu. The clock period of the system is T_{CP} = 12 tu. Because the delay of data path B is short as compared to the clock period, the clock signal that controls the latching operation of the register located between data paths A and B can be delayed by a T_{skew} of 2 tu, as shown in Fig. 6(b). This strategy delays the data signal propagating into data path B without creating any timing hazards, thereby satisfying $T_{CPB} = T_{CP} - T_{skew} \geq T_B + T_{ITB\prime}$. Alternatively, delaying the clock signal to the register, delays the latching of the data signal that propagates into data path A, adding more idle time to data path A. Therefore, both data paths have sufficient idle time, permitting the drivers to be downsized so as to reduce the power dissipation of the overall circuit. If the slower data path A is not further slowed down, the application of a non-zero clock skew provides an increased safety time to the data path A which can be used to either relax the strict timing constraints or make the circuit less sensitive to process parameter variations.

The approach presented above and illustrated in Figs. 6(a) and 6(b) provides an additional technique for saving power through the application of negative clock skew. The negative clock skew across data path A can be produced either by inserting a delay element on the clock line that distributes the clock signal to the final register of data path A, or by decreasing the transistor size of the clock buffer that drives this clock line. In the latter case, decreasing the size of the clock buffer results in less output current, and therefore, in an additional savings in power.

4. EXPERIMENTAL RESULTS FROM 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 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 20% within the data paths of certain functional unit blocks (FUBs).

A software tool has been developed to implement an optimum clock scheduling algorithm [10, 11, 12, 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 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. Using this clock period, (1), and (3), 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 of 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 value of clock skew. 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 clock scheduling tool that implements this algorithm has been applied to specific FUBs within a high performance microprocessor. A graph representing one of these FUBs is shown in Fig. 7 with normalized maximum and minimum local data path delays. All of the timing information in the analysis that follows is described in terms of these normalized path delays.

The initial clock period for the FUB shown in Fig. 7 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

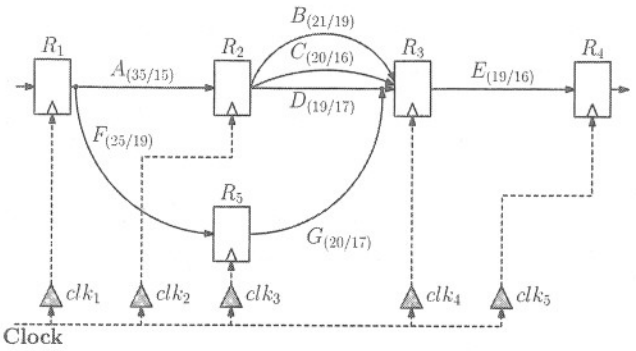


Figure 7: Circuit graph of Itanium^{TM} FUB with normalized data path delays.

circuit performance can be achieved by applying a non-zero clock skew schedule to this specific FUB.

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 in Fig. 7) that sources the clock signal that drives this register. Several different sizes of 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. 8

Normalized buffer size	Normalized clock line delay (tu)
1.00	24.93
1.43	20.14
1.71	17.79
2.05	16.27
6.07	10.90
10.47	9.67

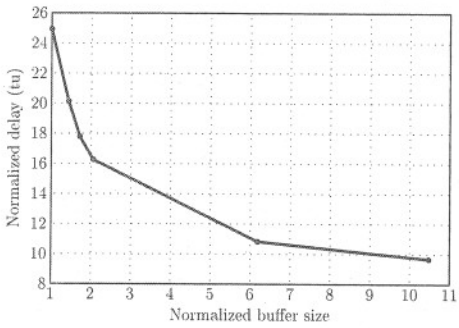


Figure 8: Variation of clock signal delay to different clock buffer sizes.

As illustrated in Fig. 8, 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 [14]. Decreasing the size of the clock buffer in order to increase the delay of the clock line achieves the objective of saving power, since the current flowing through the buffer is reduced. For the target circuit that contains the clock buffers, the power saving is approximately 1% of the total power consumed by this block.

The concept of slowing down a data path in order to save power has also been applied to certain data paths, B, C, and D, of the FUB shown in Fig. 7. Each of these data paths is slowed down by downsizing (*i.e.*, decreasing the geometric width) of the driving register R_2 by using a different register of the circuit library. The maximum and minimum delay of these data paths prior to and after decreasing the size of the data path driver is listed in Table 2.

Table 2: Comparison between the original and the increased delay of data paths B, C, and D within the FUB illustrated in Fig. 7

Data path	Original max/min data path delay (tu)	Increased max/min data path delay (tu)
B	(21/19)	(25/21)
C	(20/16)	(25/20)
D	(19/17)	(25/21)

On average, the delay of the data paths is increased by 22%. The effect of downsizing the local registers that drive the data paths is to substantially reduce the power dissipated by this circuit. The power dissipation in the circuit block containing these local registers is reduced by 82% by downsizing 69 registers. As listed in Table 2, the circuit performance can also be improved since the difference between the delay of data path A and the delays of the data paths, B, C, and D, is significant. In this case, the clock period can be reduced to $T_{CP} = 25 + \frac{35-25}{2} = 30$ tu. Therefore, the performance of the circuit can be further improved by approximately 14%.

5. CONCLUSIONS

Simulations of specific FUBs within a high performance microprocessor illustrate 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 20% by exploiting the differences in propagation delays between sequentially-adjacent data paths. A strategy for decreasing the power dissipation by slowing down the delay of the non-critical data paths has also been presented. Results demonstrate that a substantial power reduction of up to 82% in specific circuit blocks in a high performance microprocessor can be achieved by applying this strategy. A non-zero clock skew 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 both the system timing margins and power dissipation characteristics.

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, "Topological Synthesis of Clock Trees with Non-Zero Clock Skew," *Proceedings of the ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pp 158-163, December 1997.

[14] 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.