Journal of Circuits, Systems, and Computers, Vol. 11, No. 3 (2002) 231–245
© World Scientific Publishing Company

DEMONSTRATION OF SPEED AND POWER ENHANCEMENTS ON AN INDUSTRIAL CIRCUIT THROUGH APPLICATION OF CLOCK SKEW SCHEDULING

D. VELENIS[*], K. T. TANG[†,‖], I. S. KOURTEV[‡], V. ADLER[§,‖],
F. BAEZ[¶], and E. G. FRIEDMAN[*]

[*]*Department of Electrical and Computer Engineering, University of Rochester, Rochester, New York 14627, USA*
[†]*Broadcom Corporation, 2099 Gateway Place, San Jose, California 95110, USA*
[‡]*Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania 15261, USA*
[§]*Sun Microsystems, 901 San Antonio Road, Palo Alto, California 94303, USA*
[¶]*Intel Corporation, 2200 Mision College Boulevard, Santa Clara, California 95052, USA*

Received 24 April 2001
Revised 15 March 2002

A strategy to enhance the speed and power characteristics of an industrial circuit is demonstrated in this paper. It is shown that nonzero 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 nonzero clock skew scheduling algorithm is described together with a methodology that generates the required clock signal delays. Furthermore, a technique that significantly reduces the power dissipated in the noncritical data paths is demonstrated. The application of this technique combined with nonzero clock skew scheduling to the slower data paths is also described. Speed improvements of up to 18% and power savings greater than 80% are achieved in certain functional blocks of an industrial high performance microprocessor.

Keywords: Clock skew scheduling; timing margin improvement; delay management; low power timing analysis.

1. Introduction

The rapid scaling of device geometries in modern microelectronics systems supports the system-on-a-chip integration of multiple subsystems. With shrinking line dimensions and increasing chip size, 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

[‖]Drs. K. T. Tang and V. Adler contributed to the development of this project during summer internships at Intel Corporation, prior to receiving their PhD. degree from the University of Rochester.

failures. Therefore, enhanced design approaches are required to efficiently implement the clock distribution network in order to improve system performance while preventing any deleterious timing effects within the circuit.

Increasing the chip size and density adds to the on-chip power dissipation. High power dissipation penalizes the overall system since more advanced packaging and heat removal technology are necessary. Additionally, wider on-chip and off-chip power busses, larger on-chip decoupling capacitors, and more complicated power supplies are required. These factors increase the system size and cost. Furthermore, with the revolution of portable electronic devices, power dissipation has become a system performance metric, since the operation of these devices is limited by the battery life.

Design techniques and strategies to increase the speed and reduce the power dissipation have been demonstrated on an industrial circuit and are presented in this paper. To improve the circuit speed and the timing margins of a data path, nonzero clock skew scheduling has been applied to specific circuit blocks of a high performance microprocessor. In order to reduce the power dissipation, a technique that increases the delay of the noncritical data paths to exploit power savings has been applied to this circuit.

This paper is organized as follows. The application of nonzero clock skew scheduling to increase the speed and enhance the performance of a circuit is presented in Sec. 2. A strategy that reduces the power dissipation by delaying the noncritical data paths while exploiting nonzero clock skew is discussed and demonstrated in Sec. 3. Finally, some conclusions are presented in Sec. 4.

2. Speed Enhancements

Many of the techniques that have been developed to improve the 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 can be 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$^{\text{TM}}$,[a] the first processor in the Intel's IA-64 microarchitecture family.[6,7] In this section, the effectiveness of the application of nonzero clock skew scheduling to improve performance and minimize the likelihood of race conditions is demonstrated. Background information characterizing clock skew is presented in Sec. 2.1. An algorithm to implement a nonzero clock skew schedule is discussed in Sec. 2.2.

[a]Itanium$^{\text{TM}}$ is a registered trademark of Intel Corporation.

Finally, a demonstration of the application of this technique on certain blocks of an industrial high performance microprocessor is presented in Sec. 2.3.

2.1. *Background*

A synchronous digital 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.

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 $T_{\text{skew}} = T_{CD_i} - T_{CD_j}$, 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]

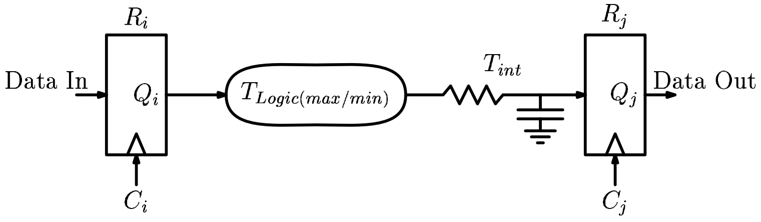


Fig. 1. A local data path.

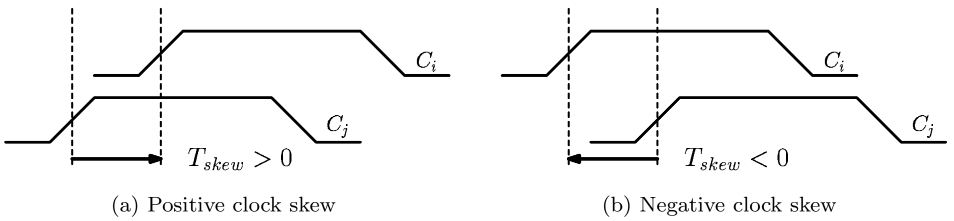


Fig. 2. Examples of positive and negative clock skew.

The strategy of minimizing clock skew has been a central design technique for decades in synchronous digital 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 Ref. 8 further optimization of the circuit performance and reliability can be achieved by applying nonzero 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.

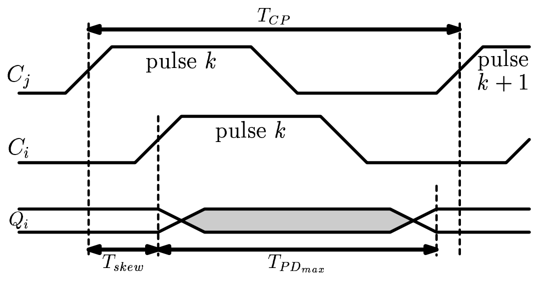
Table 1. Timing relationships for a local data path R_i to R_j.

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

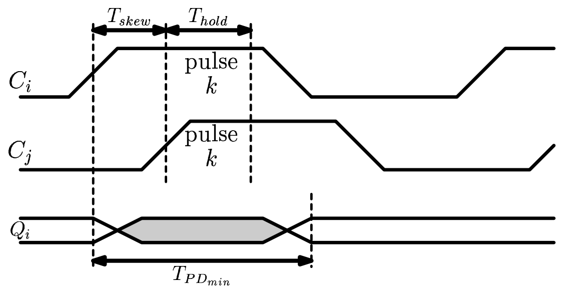
In the inequalities listed in Table 1, T_{skew} is the clock skew between registers R_i and R_j. $T_{PD_{\max}}$ ($T_{PD_{\min}}$) is the maximum (minimum) propagation delay between registers, R_i and R_j, shown in Eqs. (3) and (4), respectively. $T_{\text{Logic(max)}}$ ($T_{\text{Logic(min)}}$) is the 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. $T_{\text{set-up}}$ is the time required for the data signal 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.

From the inequalities listed in Table 1, Eq. (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, Eq. (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*.[1,8,11,12] Timing relationships that prevent zero and double clocking are shown in Figs. 3(a) and 3(b), respectively.

For a given clock period T_{CP}, Eqs. (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*[10,13] between the minimum permissible clock skew $T_{\text{skew(min)}}$ and the maximum permissible clock skew $T_{\text{skew(max)}}$. The permissible



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



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

Fig. 3. Timing hazards in synchronous digital systems.

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. $T_{\text{skew(max)}}$ 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.

2.2. *Algorithm implementation*

The optimal clock scheduling problem has been described in Ref. 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 Refs. 10 and 13. These concepts have been further enhanced, implemented as an algorithm, integrated into a software tool,[11,12,14] and applied to a functional unit within a high performance microprocessor to determine an optimal clock skew schedule.

The development of a software tool to implement this optimum clock skew scheduling algorithm is described in Refs. 10 to 13. The input data to this tool are the minimum and maximum delays of each of the local data paths of the circuit. 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:

(i) 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.
(ii) 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.
(iii) Using the clock period specified from step 2, the permissible clock skew range is calculated from Eqs. (1) and (2) for each pair of sequentially-adjacent registers in C.
(iv) 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 skew schedule exists for the clock period specified in step 2.
(v) If a feasible clock skew 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 nonfeasible clock skew 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 (ii) and (v) 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, Eqs. (1) and (2), the clock skew between each pair of sequentially-adjacent registers within C is computed.

(vi) 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 Eqs. (1) and (2). The choice of ε is a tradeoff between performance and the computational run time of the algorithm.

2.3. *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 nonzero 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 nonzero clock skew scheduling to these circuits yields a speed improvement of up to 18% within the data paths of certain functional unit blocks (FUBs).

The clock scheduling tool described in Sec. 2.2 and in Refs. 11 and 12 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

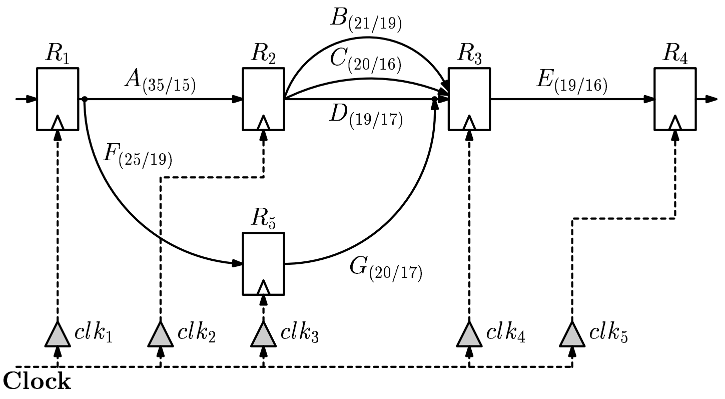


Fig. 4. Circuit graph of Itanium™ FUB with normalized data path delays.

signal to propagate along data path A is $T_{CP} + T_{\text{skew}} = 28 + 7 = 35$ 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 nonzero clock skew schedule to this specific FUB.

The added delay to the path from the clock source to register R_2 is achieved by decreasing the size of the clock buffer (clk_2 shown in Fig. 4) that sources the clock signal that drives the register. This delay change is accomplished by replacing the clock buffer with a 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 Table 2 and illustrated in Fig. 5

Table 2. Alternative buffer sizes and buffer delays for register R_2.

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

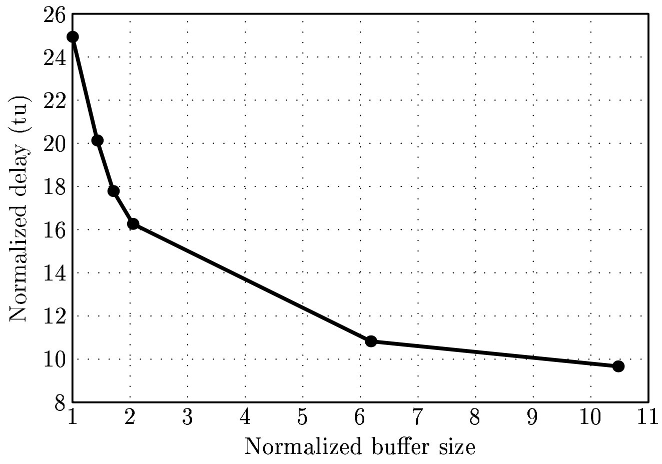


Fig. 5. Variation of clock signal delay to different clock buffer sizes.

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 Table 2). In order to produce an additional clock delay of 7 tu to drive register R_2, buffer No. 4, is used instead. The 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 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 beneficial 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.

3. Reducing Power in Noncritical Data Paths

Two of the most popular techniques that are used to reduce power dissipation are supply voltage (V_{dd}) scaling and clock gating.[17,18] V_{dd} reduction is an effective way for reducing power, since power dissipation is proportional to the square of V_{dd}. The disadvantages of supply voltage scaling are effects such as sub-threshold and gate oxide leakage and increased sensitivity to noise.[19] Clock gating reduces the capacitance being switched by the clock distribution network.[18] The major disadvantages of clock gating are the increased complexity of the timing analysis and the increased transient currents when large blocks of logic are switched on and off.

Another technique to reduce the dissipated power is the use of smaller size circuit elements from predesigned cell libraries in order to achieve significant power savings. The smaller sized elements introduce smaller load capacitances albeit with a small delay penalty.[17] When this technique is applied to noncritical data paths, the delay penalty has no impact on the overall performance of the system. A demonstration of the application of this technique to an industrial circuit is presented in this section. It is shown that significant improvements in power dissipation can be achieved. Additionally, a methodology to expand this technique to slower (more critical) data paths is also discussed in this section.

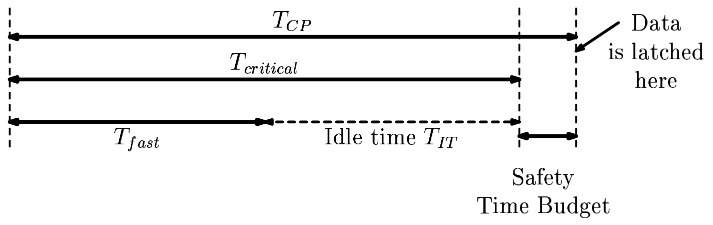
The concept of the technique, the delay constraints, and the limitations in power savings are presented in Sec. 3.1. The necessary conditions to apply this technique to slower data paths are described in Sec. 3.2. Simulation results that demonstrate the power savings achieved on an industrial circuit are presented in Sec. 3.3.

3.1. *The general technique and related delay constraints*

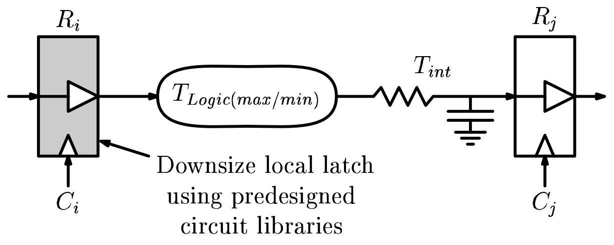
In a large high performance synchronous digital system, such as a microprocessor, the number of critical data paths is small as compared with the total number of data paths in the system. For example, in a specific system described in Ref. 11, less than 5% of the total data paths are within 20% of the maximum path delay while more than 65% of the total data paths have path delays less than half of the maximum 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 fast 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. 6(a). 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 (i.e., decreasing the geometric width) of the latch R_i that drives the data path as shown in Fig. 6(b), using smaller sized circuits from a predesigned cell library. By downsizing the latch the effective capacitance of the latch is decreased and the power required to drive the latch is reduced. Also, the geometric width of the output driver within the latch is decreased, thereby reducing the output current of the latch[16] and increasing the path delay. Therefore, this procedure results in a decrease in power consumption, albeit with an increase in the data path delay.

There are constraints, however, that limit the minimum size of an output driver and thereby the additional delay that can be introduced. One constraint is that the additional delay should not exceed the maximum permissible path delay constraint



(a) Short data path delay as compared to a critical long data path delay



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

Fig. 6. Increasing the delay of the fast data paths by downsizing the local latches that drive these paths.

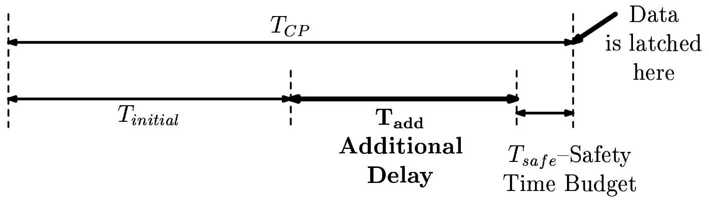


Fig. 7. The added delay of the fast data path should not violate the long path timing constraint.

as shown in Fig. 7. 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 limit, equal) to the clock period T_{CP}.

Another constraint is that the introduction of smaller sized output drivers should not degrade the signal rise and fall times below some target level. Due to the reduced size of the output driver, the output signal transition time of the latch is slower, increasing the short-circuit power dissipation on the gates that are driven by the latch. The short-circuit power dissipation is due to the current that flows directly from the power supply to the ground of a CMOS gate when the input voltage is within the range V_{tn} and $V_{dd} + V_{tp}$ (when both the PMOS and NMOS transistors are on). When the transition time of the input voltage is longer, the time during which both transistors are on is also longer, increasing the short-circuit power dissipation. A close approximation of the short-circuit power dissipation is given by[20]:

$$P_{SC} = \frac{1}{2} I_{\text{peak}} t_{\text{base}} V_{dd} f \,, \tag{5}$$

where I_{peak} depends on the size of the transistors of the driven gate, t_{base} is the input signal transition time, and f is the switching frequency of the input signal.

As shown by Eq. (5), as the size of the output buffer of the latch is decreased, the input signal transition time t_{base} increases, increasing the short-circuit power dissipated in the load gates. Therefore, there is a lower limit on decreasing the size of the output driver to achieve less power.

3.2. *Application to critical data paths*

The concept of slowing down fast data paths in order to save power can be further applied to slower, more critical data paths with the aid of *nonzero clock skew scheduling.*[8,11] By applying negative clock skew to the slower, more critical data paths, the idle time in these data paths can be increased, permitting these paths to be slowed down further. 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 satisfy the zero clocking timing constraint.

An example of the application of this concept to long data path delays is shown in Fig. 8. As shown in Fig. 8(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 2 tu, as shown in Fig. 8(b). This strategy delays the data signal propagating into data path B without creating any timing hazards, satisfying $T_{CPB} = T_{CP} - T_{\text{skew}} \geq T_B + T_{ITB'}$. Alternatively, delaying the arrival of the clock signal at 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 slow data path A is not further slowed down, the application of negative clock skew can increase the safety margin of data path A, which can be used to relax the strict timing constraints and make the circuit less sensitive to process parameter variations.[13]

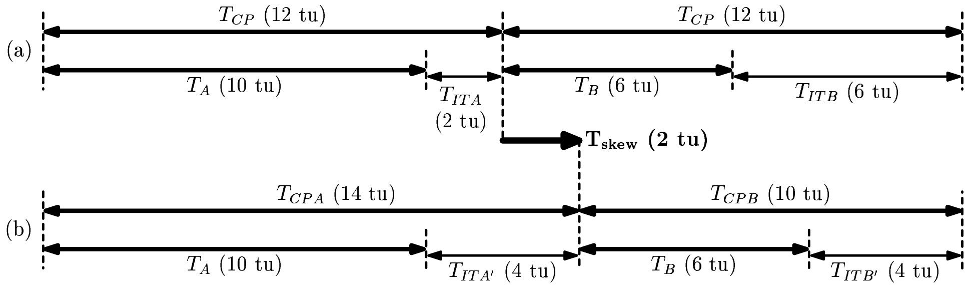


Fig. 8. Application of local clock skew to equalize the available idle time between the long and short delay data paths. (a) Initial timing of the data paths. (b) Timing of the data paths after the application of local clock skew.

The approach presented above and illustrated in Figs. 8(a) and 8(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 along the clock signal path 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, which provides an additional savings in power.

3.3. *Results on a demonstration circuit*

The efficiency of this technique to decrease the power dissipation of noncritical data paths by changing the timing of these paths has been demonstrated on certain FUBs of a high performance industrial microprocessor. One of these FUBs is illustrated in Fig. 4.

Table 3. Comparison between the original and the increased delay of data paths B, C, and D within the FUB illustrated in Fig. 4.

Data path	Original max/min data path delay (tu)	Increased max/min data path delay (tu)	Increased delay (%)
B	(21/19)	(25/21)	14.7
C	(20/16)	(25/20)	22.5
D	(19/17)	(25/21)	27.5
	Average increase in delay (%)		21.6

Table 4. Normalized power dissipation within the circuit block containing the latches.

No optimization	Downsize latches w/o clock scheduling	Downsize latches w. clock scheduling
100%	18%	17.3%

The technique has been applied to the fast data paths, B, C, and D, of the FUB shown in Fig. 4. Each of these data paths is slowed down by downsizing the driving latch R_2 by using a different latch selected from a 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 3. It is shown in Table 3 that the delay of these data paths is increased on average by 21.6%.

The effect of downsizing the local latches that drive the data paths is to substantially reduce the power dissipated within the circuit block that contains these latches. As shown in Table 4, the total power dissipation of the circuit block is reduced by 82% by downsizing a total of 69 latches.

The remaining data paths within the FUB are unchanged. The effect of changing the latch on the data signal rise and fall times in data paths B, C, and D is negligible. Also, no maximum data path delay constraint is violated since the larger maximum delay of the affected data paths (25 tu) is less than the maximum delay of the most critical data path (35 tu). Since the difference between the delay of data path A and the delays of the data paths, B, C, and D, is significant, the circuit performance can be improved with the application of nonzero clock skew scheduling, as described in Sec. 2. In this case, the clock period can be reduced to $T_{CP} = 25 + \frac{35-25}{2} = 30$ tu. The performance of the circuit can therefore be further enhanced by approximately 14%. Furthermore, the application of negative clock skew to downsize clock buffers results in an additional 4% decrease of the power dissipation as shown in Table 4. This decrease is due to the reduced capacitance of the clock buffer that drives the downsized latches.

4. Conclusions

Simulations of specific FUBs within a high performance commercial microprocessor demonstrate that improvements in the timing margin of the data paths can be achieved by applying nonzero clock skew. It is shown that in specific circuit blocks the timing margins can be increased by up to 18% by exploiting the differences in propagation delays between sequentially-adjacent data paths. The required clock delays from the clock source 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 nonzero clock skew scheduling software tool has also been developed.[11,14] This tool has been evaluated on numerous industrial circuits,[11] demonstrating the general utility of clock skew scheduling to improve the timing characteristics of a synchronous digital system.

A strategy for decreasing the power dissipation by reducing the size of the driving latches and increasing the delay of the noncritical data paths has also been demonstrated. The constraints, advantages, and disadvantages have been discussed. The application of nonzero clock skew scheduling to increase the idle time of the slower data paths has also been presented. Simulations on specific functional unit blocks within a high performance industrial microprocessor demonstrate that a substantial local power reduction of greater than 80% can be achieved by applying this strategy.

Acknowledgment

This research was supported by a grant from Intel Corporation.

References

1. E. G. Friedman, *Clock Distribution Networks in VLSI Circuits and Systems*, IEEE Press, Piscataway, New Jersey, 1995.
2. E. G. Friedman and S. Powell, "Design and analysis for a hierarchical clock distribution system for synchronous standard cell/macrocell VLSI", *IEEE J. Solid-State Circuits* **SC-21**, 2 (1986) 240–246.
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", *Proc. IEEE Int. Conf. Computer Design*, October 1986, pp. 118–122.
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 Trans. Circuits Syst. II: Analog and Digital Signal Processing* **39**, 11 (1992) 799–814.
5. A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Kluwer Academic Publishers, Boston, Massachusetts, 1995.
6. U. Desai, S. Tam, R. Kim, and J. Zhang, "Itanium processor clock design", *Proc. ACM/SIGDA Int. Symp. Physical Design*, April 2000, pp. 94–98.
7. S. Rusu and S. Tam, "Clock generation and distribution for the first IA-64 microprocessor", *Proc. IEEE Int. Solid State Circuits Conference*, February 2000, pp. 176–177.
8. J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Comput.* **39**, 7 (1990) 945–951.
9. J. L. Neves and E. G. Friedman, "Design methodology for synthesizing clock distribution networks exploiting nonzero clock skew", *IEEE Trans. VLSI Syst.* **VLSI-4**, 2 (1996) 286–291.

10. J. L. Neves and E. G. Friedman, "Buffered clock tree synthesis with nonzero clock skew scheduling for increased tolerance to process parameter variations", *J. VLSI Signal Processing* **16**, 2/3 (1997) 149–161.
11. I. S. Kourtev and E. G. Friedman, *Timing Optimization Through Clock Skew Scheduling*, Kluwer Academic Publishers, Norwell, Massachusetts, 2000.
12. I. S. Kourtev and E. G. Friedman, "Synthesis of clock tree topologies to implement nonzero skew schedule", *IEE Proc. Circuits, Devices and Syst.* **146**, 6 (1999) 321–326.
13. J. L. Neves and E. G. Friedman, "Optimal clock skew scheduling tolerant to process variations", *Proc. ACM/IEEE Design Automation Conference*, June 1996, pp. 623–628.
14. I. S. Kourtev and E. G. Friedman, "Clock skew scheduling for improved reliability via quadratic programming", *Proc. IEEE Int. Conf. Computer-Aided Design*, November 1999, pp. 239–243.
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 nonzero clock skew scheduling", *Proc. ACM/IEEE Int. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, December 2000, pp. 58–63.
16. V. Adler and E. G. Friedman, "Repeater design to reduce delay and power in resistive interconnect", *IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing* **CAS II-45**, 5 (1998) 607–616.
17. V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing power in high-performance microprocessors", *Proc. IEEE/ACM Design Automation Conference*, June 1998, pp. 732–737.
18. L. Benini, P. Siegel, and G. De Micheli, "Saving power by synthesizing gated clocks for sequential circuits", *IEEE Design & Test of Computers* **11**, 4 (1994) 32–41.
19. K. Chen and C. Hu, "Performance and V_{dd} scaling in deep submicrometer CMOS", *IEEE J. Solid-State Circuits* **SC-33**, 10 (1998) 1586–1589.
20. V. Adler and E. G. Friedman, "Delay and power expressions for a CMOS inverter driving a resistive-capacitive load", *Analog Integrated Circuits and Signal Processing* **14**, 1/2 (1997) 29–39.