INT. J. ELECTRONICS, 1991, VOL. 70, NO. 5, 917–935

Pipelining of high performance synchronous digital systems

EBY G. FRIEDMAN† and J. H. MULLIGAN, JR‡

A new approach is described to the design of those synchronous digital systems in which the performance parameters latency and clock frequency are of primary importance. Specifically, the trade-off between clock frequency and latency is analysed in terms of the circuit characteristics of a pipelined data path. A design paradigm relating latency and clock frequency as a function of the level of pipelining is described for studying the performance of a synchronous system. This perspective permits the development of design equations for constrained and unconstrained design problems from which the optimal level of pipelining can be determined in terms of the delays of logic, interconnect, and registers, and the clock skew and number of logic stages.

1. Introduction

In the design of high performance synchronous digital systems, such as radar, sonar, and many types of high speed computers, there is a considerable desire for maximum performance. Performance, however, can be defined in many ways. Two key measures of performance are the latency of the system, i.e. the total time required to move a particular signal from the input of a system to its output, and the maximum clock frequency, which is measured by how often new data appear at the output of a synchronous digital system. In this paper, performance is defined solely in terms of time and is represented by the latency and the clock frequency of the system. Thus, characteristics such as area or power dissipation are viewed as secondary design objectives.

This paper extends results previously reported by the authors (Friedman and Mulligan 1991); for the convenience of the reader, relevant portions of that paper are reviewed, thus permitting this one to be self-contained. This paper provides additional insight into the general use of the earlier results in the exploration of speed/area/power trade-offs within the system design space and examples are provided which illustrate the use of these results. Finally, additional interpretations and derivations are presented to facilitate the application of these results to system design.

In digital systems, the minimum latency occurs when the data path consists entirely of logic stages; it is the time required for propagation of a data signal through this logic path. The clock period for this system, which is also the latency, is equal to the time required to process one data sample. If the time interval at which new data appear at the input of a system is smaller than the latency, for this simple configuration, registers can be inserted into the data path to increase the frequency at which new data signals can be processed through the system and appear at the system output. This degrades the latency, however. The process of

Received 23 August 1990; accepted 3 September 1990.

† Hughes Aircraft Company, 6155 El Camino Real, Carlsbad, CA 92009, U.S.A.

‡ Department of Electrical and Computer Engineering, University of California, Irvine, CA 92717, U.S.A.

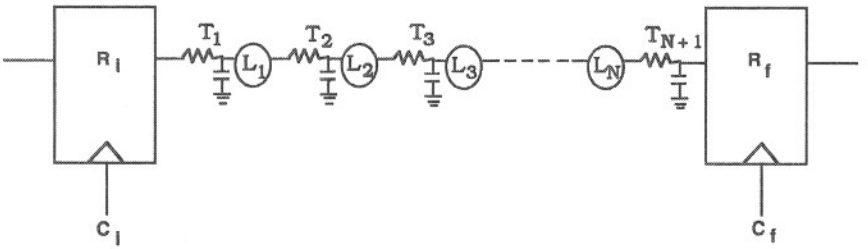

Figure 1. Synchronous data path with N stages of logic.

inserting registers into a data path to increase the system clock frequency is spoken of as pipelining.

This paper deals with the design of systems in which one desires to maximize the clock frequency, minimize the latency, or provide a trade-off solution between minimum latency and maximum clock frequency. The paper consists of four principal sections. The relative timing of the clock signals synchronizing the data path is described in § 2. A graphical interpretation of the performance trade-offs of a pipelined feed-forward non-recursive system is presented in § 3 with its supporting design equations, illustrating the constraints, limitations and trade-offs within the design space of a synchronous digital system.

Specific performance requirements such as minimum clock frequency or maximum latency are common to most high performance synchronous digital systems. In these systems, the design problem is either one of maximizing the clock frequency while not exceeding a maximum latency or minimizing the latency while meeting a specified clock frequency. In other less aggressive applications, neither the latency nor the clock frequency ultimately constrains the design problem. In these unconstrained design problems, the degree of pipelining of the data path can be chosen to trade-off the latency with the clock frequency. These constrained and unconstrained systems are discussed in § 4. This design paradigm also supports the optimization of other performance parameters, such as area and power, when the system is unconstrained by the latency and the clock frequency requirements. In § 5, approaches are described to integrate these additional design requirements, from the context of the design paradigm, into the system implementation. Some concluding remarks are presented in § 6.

2. Clock distribution networks

The times of arrival of the initial clock signal C_i and the final clock signal C_f shown in Fig. 1 define the time reference when the data signals begin to leave their respective registers. These clock signals originate from a clock distribution network which is typically designed to generate a specific clock signal waveform which synchronizes each register (Friedman and Powell 1986, Hatamian and Cash 1987, Hatamian 1988). The difference in delay between two sequentially adjacent clock paths is described as the clock skew, T_{SKEW}. If the clock signals C_i and C_f are in complete synchronism (i.e. the clock signals arrive at their respective registers at exactly the same time), the clock skew is zero. If the time of arrival of the clock signal at the final register of a data path, C_f, leads that of the time of arrival of the clock signal at the initial register of the same sequential data path, C_i, the clock

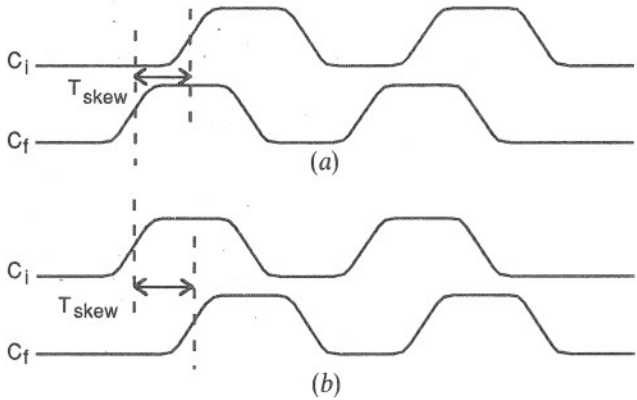

Figure 2. Clock timing diagrams: (*a*) positive clock skew; (*b*) negative clock skew.

skew is defined as positive (see Fig. 2(*a*)) and, under this condition, the maximum attainable operating frequency is decreased. Positive clock skew is the additional amount of time which must be added to the minimum clock period to apply a new clock signal at the final register reliably, where reliable operation implies that the system will function correctly at low as well as high frequencies.

If C_f lags C_i, the clock skew is defined to be negative (see Fig. 2(*b*)): negative clock skew can be used to improve the maximum performance of a synchronous system by decreasing the delay of a critical path. This negative clock skew represents the additional amount of time which the data signal at R_i has to propagate through the N stages of logic and $N+1$ sections of interconnect and into the final register. This clock skew subtracts from the logic path delay, thereby decreasing the minimum clock period. The maximum permissible negative clock skew of any data path, however, is dependent upon the magnitude of the clock period itself as well as the previous data paths. This results from the structure of the serially cascaded global data path. Since a particular clock signal synchronizes a register which functions in a dual role—as the initial register of the next local data path and as the final register of the previous data path—the earlier C_i is for a given data path, the earlier that same clock signal, now C_f, is for the previous data path. Thus, for a particular clock signal, a large negative clock skew when behaving as C_i would make the clock skew of the previous data path, in which the clock signal is now behaving as C_f, that much more positive. It should be noted that Hatamian and Cash (1987) and Hatamian (1988) have described many of these characteristics of clock skew and their effects on the maximum clock frequency and designate the lead/lag clock skew polarity (positive/negative clock skew) opposite to that described herein.

3. Design paradigm for pipelined synchronous systems

When registers are inserted into the path from the system input to its output (defined as the global data path), the minimum clock period can be decreased (providing a higher maximum clock frequency), albeit with an increase in latency. Each global data path is therefore composed of individual cascaded register-to-register data paths and these are defined as local data paths. Each local data path is composed of an initial and final register and typically, n logic stages between

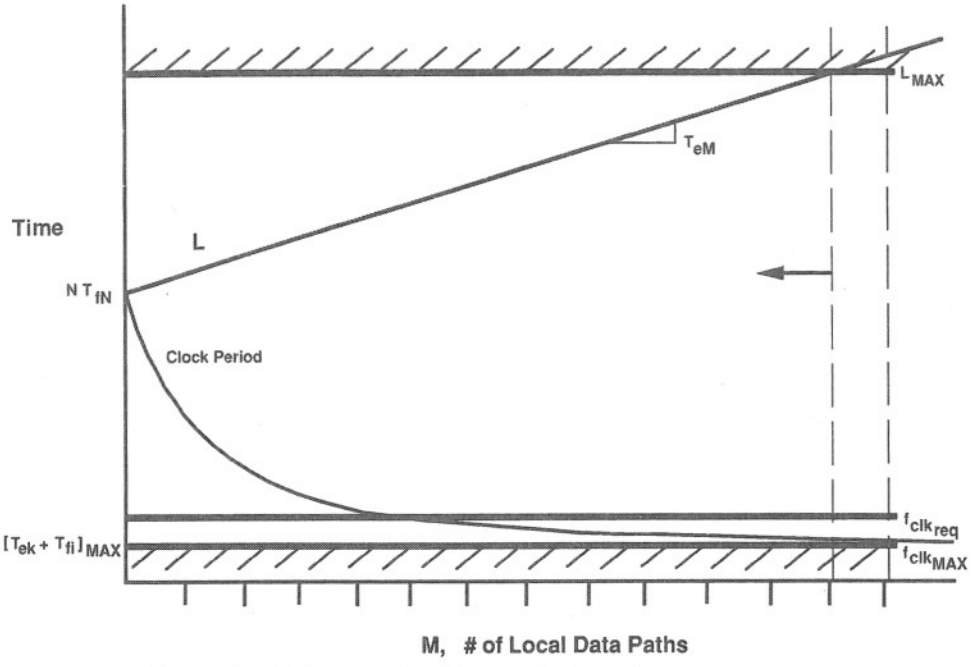

Figure 3. Design paradigm for pipelined synchronous systems.

them. Note that each register within a local data path performs double duty, serving as the initial (final) and final (initial) register of the current and previous (next) local data path, respectively.

The trade-off between clock frequency and latency is depicted in Fig. 3. In this figure, both the latency and the clock period are shown as a function of the number of pipeline registers M inserted into a global data path. As M increases, the latency increases for each inserted register and the maximum possible clock frequency increases. This occurs because the critical path is shortened since there are less logic and interconnect stages per local data path.

If no registers are inserted into the data path, the minimum latency $L_{\min}$ is the summation of the individual logic delays, NT_{fN}, as shown by (1) and depicted in Fig. 3 as both the latency and the clock period when M equals zero. Therefore, for N logic stages, $L_{\min}$ can be expressed as

$$L_{\min} = \sum_{i=1}^{N} T_{fi} = NT_{fN} \tag{1}$$

where the individual logic stage delay T_{fi} and the average logic stage delay T_{fN} are defined in the Appendix.

The maximum clock frequency at which a synchronous digital system can move data is given by (2) below:

$$f_{clk} = \frac{1}{T_{cp}} \leqslant \frac{1}{T_{PD} + T_{SKEW}} \tag{2}$$

where T_{cp} is the clock period, T_{PD} is defined in (A 3) in the Appendix, and the local

data path with the greatest $T_{PD}+T_{SKEW}$ represents the critical path of the system. Note that for positive clock skew, the maximum clock frequency decreases, while for negative clock skew the maximum clock frequency increases.

The latency L is defined as the time required to move a data signal from the input of the system to its output. For the special case of no pipelining, the latency equals the maximum clock period. If a single register is inserted into the data path, registers external to the system are required to synchronize the external data flow of the signal path. Two registers, one at the input and the other at the output of the global data path, represent a self-contained synchronous system (as shown in Fig. 1). Each additional register increases the latency and decreases the minimum clock period. Thus, the latency of a pipelined data path is the summation of the total delay through the global data path as shown below in (3) and (4).

$$L=\sum_{i=1}^{N} T_{fi}+\sum_{k=1}^{M} T_{ek} \tag{3}$$

$$L=NT_{fN}+MT_{eM} \tag{4}$$

where N is the number of logic stages per global data path, M is the number of local data paths (and clock distribution networks) per global data path, $M+1$ is the number of clock periods (and registers) required to move a particular data signal from the input of the system to its output, and the maximum permissible negative clock skew T_e in (3) and (4) can be represented by (5). In this equation,

$$T_e=T_{REG}+T_{SKEW} \tag{5}$$

is the aggregate delay due to the initial and final registers T_{REG} and the clock distribution network of each local data path T_{SKEW}. T_e can be used to represent the margin of error or the acceptable tolerance of negative clock skew for each local data path. Note that when T_{SKEW} is zero, T_e equals T_{REG}, where T_{REG} is defined in (A 2). Also note that T_e is typically positive for most circuit configurations.

T_e can be described as the total effective delay of the registers and clock distribution network per local data path. Each local data path within a global data path provides its own T_{ek} where k is the kth local data path. The average T_{ek} for a global data path over all the M serially connected cascaded data paths is defined as T_{eM}. Since T_{fN} is the average delay of each of the logic and interconnect stages per data path, (4) describes the latency in terms of average delays instead of individual summations for convenience and improved interpretation.

The average number of logic stages per local data path n is given by (6).

$$n=\frac{N}{M} \tag{6}$$

The clock period T_{cp} can be expressed as

$$T_{cp}\geqslant T_{REG}+nT_{fN}+T_{SKEW} \tag{7}$$

$$T_{cp} = \begin{cases} NT_{fN} & \text{for } M = 0 \quad (8) \\ T_{eM} + \dfrac{NT_{fN}}{M} & \text{for } M \geqslant 1 \quad (9) \end{cases}$$

For a pipelined global data path with registers placed at both its input and output, L can also be described by the relation

$$L = \frac{M+1}{f_{clk}} \tag{10}$$

By substituting (2),(5) and (7) into (10), the total latency of a partitioned global data path can be expressed in terms of the average delay components of a local data path as

$$L = [M+1] \cdot [nT_{fN} + T_{eM}] \tag{11}$$

These results assume that a global data path is partitioned into local data paths of approximately equal delay. By applying the concept of negative clock skew, one can, in effect, even out the delays of each pipelined data path, making all local data paths have approximately equal delays.

Thus, from (4), as each additional register is inserted into the global data path, L increases by T_{eM}. Thus, L increases linearly with M as shown in (4) and depicted in Fig. 3. As shown in (9) and Fig. 3, the clock period exhibits an inverse proportionality to M. As M is increased, the maximum practical clock frequency is reached when n equals one, as defined by $T_{ek} + T_{fi}$. This assumes that the circuit is not a simple shift register and logical operations are being performed between the registers of each local data path (if the global data path is a simple shift register, n equals zero and the maximum clock frequency is limited by the worst case T_{ek}). The MAX subscript used in Fig. 3 is intended to emphasize that the critical local data path limits the minimum clock period (or maximum clock frequency) of the total global data path.

Most design requirements must satisfy some specified maximum time for latency while satisfying or surpassing a required clock frequency. The design constraints due to an application specific limitation on the maximum permissible latency L_{max} and the maximum possible clock frequency $f_{clk_{MAX}}$ are shown in Fig. 3 by the vertical dashed lines. From L_{max}, an appropriate maximumn clock frequency and level of pipelining M is defined by the intersection of the L curve and the L_{max} line. If L_{max} is not specified or is very large and the desire is to make the clock frequency as high as possible, then an appropriate f_{clk} is defined by the intersection of the clock period curve and the $f_{clk_{MAX}}$ line. Thus, for a particular L and f_{clk}, the possible design space is indicated by the horizontal arrow. If L and f_{clk} are both of importance and no L_{max} or $f_{clk_{MAX}}$ is specified or constrains the design space, then some optimal level of pipelining is required to provide a 'reasonably high' clock frequency while maintaining a 'reasonable' latency. This design choice is represented by a particular value of M which defines an application specific f_{clk} and L.

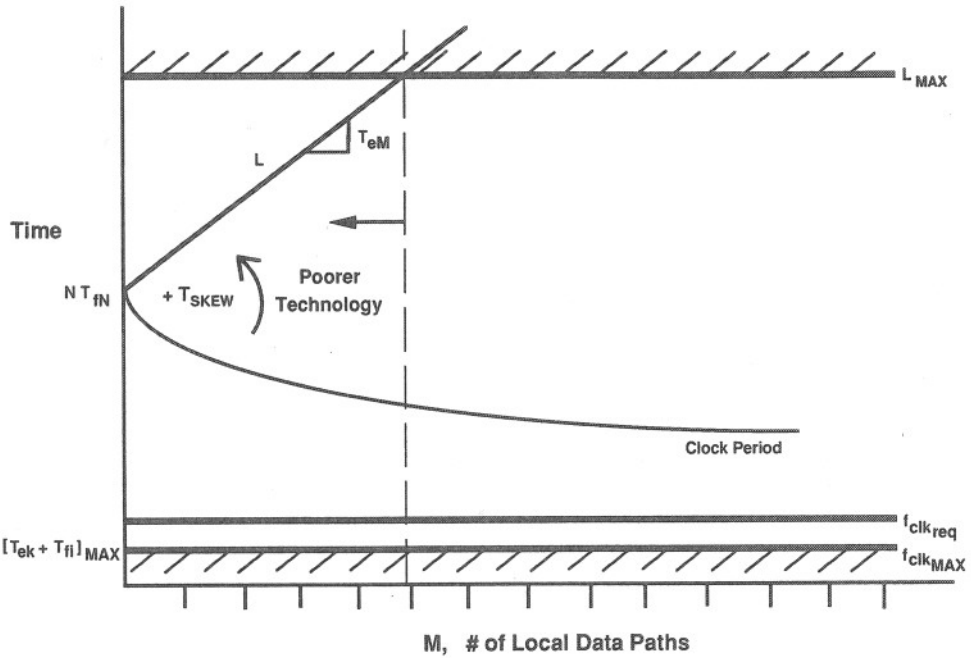

Figure 4. Effect of positive clock skew and technology on design paradigm.

The effects of clock skew, technology, and logic architecture on clock period and latency are graphically demonstrated in Figs 4 and 5. If the clock skew is positive or if a poorer (i.e. slower) technology is used, then, as shown in Fig. 4, T_{eM} increases and L quickly reaches $L_{\max}$. Also, the minimum clock period increases which decreases the maximum clock frequency and which, for large positive clock skew or a very poor technology, eliminates any possibility of satisfying a specified clock frequency $f_{\text{clk}_{\text{req}}}$ and limits the total design space as defined by the intersection of L and $L_{\max}$. In addition, for a poorer technology or logic architecture, the intersection between either the clock period or the latency curve and the ordinate shifts upwards since T_{fN} increases owing to the slower technology and N increases for the less optimal architecture.

If the clock skew is negative or a better (i.e. faster) technology is used, as shown in Fig. 5, T_{eM} decreases, permitting the latency to be less dependent on M. Also, the minimum clock period decreases, satisfying $f_{\text{clk}_{\text{req}}}$ more easily and $f_{\text{clk}_{MAX}}$ with minimal pipelining. The possible design space, represented by the intersection of L and $L_{\max}$, is greatly increased, permitting higher levels of pipelining if very high clock rates are desired. In addition, for a better technology or logic architecture, the intersection between either the clock period or the latency curve and the ordinate shifts downward since T_{fN} decreases owing to the faster technology and N decreases for a more optimal architecture.

Thus, Figs 4 and 5 graphically describe how clock skew, technology, and logic architecture affect both the latency and the maximum clock frequency of a pipelined synchronous digital system. Finally, for applications which are not limited by the maximum latency or required clock frequency, it is shown in § 5 how this design paradigm can be used to optimize performance parameters such as

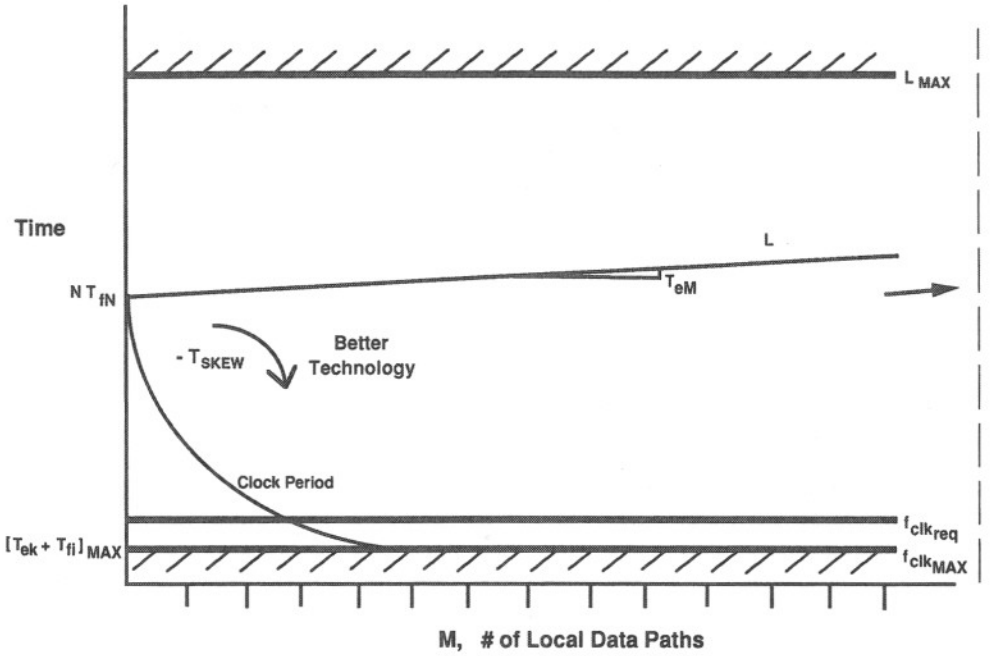

Figure 5. Effect of negative clock skew and technology on design paradigm.

area and power dissipation while still satisfying the system specified clock frequency and latency requirements.

4. Design objectives

Three types of design problems can be considered using this approach:

(*a*) the maximum latency constrains the design problem;

(*b*) the required clock frequency constrains the design problem;

(*c*) the problem is unconstrained and a trade-off between L and f_{clk} must be made.

4.1. *Maximum latency*

In applications where the maximum latency of a system is specified and $L_{\max}$ constrains the design space, the degree of pipelining can be determined from (12), where T_{eM} is the average $T_{\text{REG}}+T_{\text{SKEW}}$ delay of all of the M pipelined data paths of an N stage global data path,

$$M \leqslant \frac{L_{\max}-NT_{fN}}{T_{eM}} \tag{12}$$

A range of possible values of clock frequency is given in (13), where the value of M is determined from (12). The lower bound on clock frequency is due to the constraint on maximum latency and the upper bound is required to ensure that the appropriate data for the time period are latched into the register.

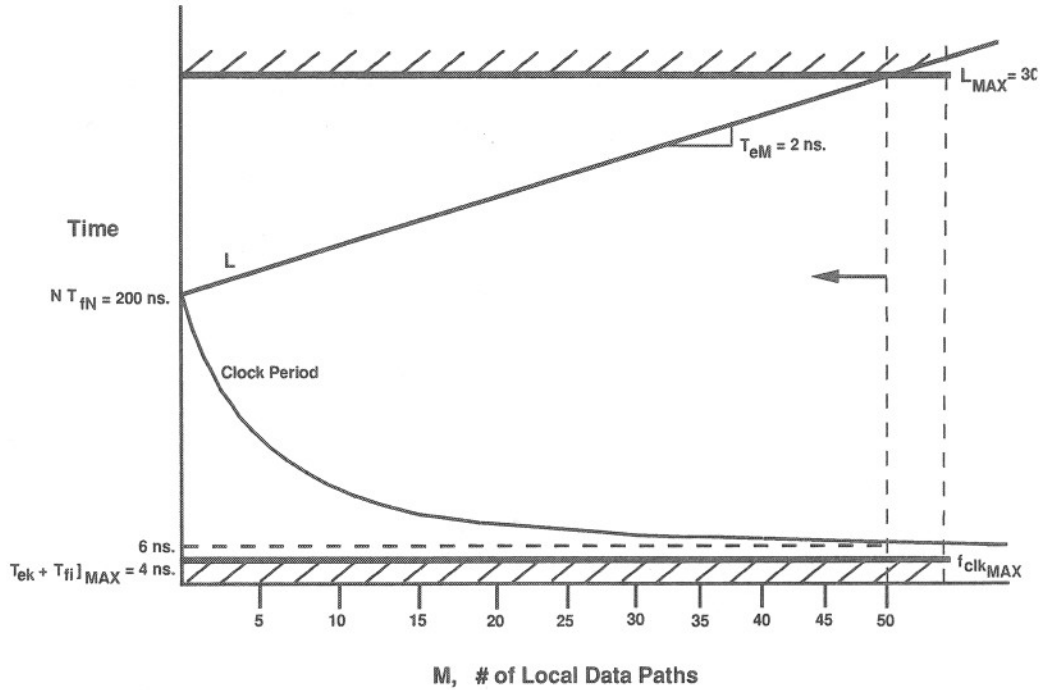

Figure 6. Example of design paradigm with constraining maximum latency.

$$\frac{M}{L_{\max}} \leqslant f_{\text{clk}} \leqslant \frac{M}{MT_{eM} + NT_{fN}} \tag{13}$$

Thus, as shown in Fig. 3, for given maximum latency and knowledge of the average logic, register, and clock delay characteristics of a global data path, the degree of pipelining and range of clock frequency can be determined directly. An example is provided below describing this latency constrained design problem.

Example 1: *Determining clock frequency for a specified latency*

This example assumes that the delay characteristics of a pipelined data path are known and the focus of the problem is to determine the range of frequency at which a system should be clocked while not exceeding a specified latency goal. This example can be explained in the context of Fig. 3 where $L_{\max}$ cannot be exceeded while providing as high a clock frequency as possible. Thus, the appropriate level of pipelining to maximize f_{clk} while satisfying the constraint on L is determined by the intersection of the L and $L_{\max}$ curves, defining both M and f_{clk}.

Equations (12) and (13) can be used to determine the appropriate number of registers and the range of possible clock frequency, respectively, where it is noted that T_{eM} is the average $T_{\text{REG}} + T_{\text{SKEW}}$ of each local data path along the global pipelined data path. Thus, for a 100 stage data path where the average stage delay T_{fN} and average register and skew delay T_{eM} are both 2 ns, f_{clk} and an upper limit for M can be directly determined for a given target L as shown in Fig. 6. If L must be less than 300 ns, then the maximum number of local data paths (M),

corresponding to the upper limit given by (12), is 50 (requiring 51 pipeline registers). Thus, two logic stages per local data path, $n=2$, are appropriate for this 100 stage system. In order not to exceed the target latency of 300 ns, the data must flow from register to register a maximum of every 6 ns (T_e+2T_{fN}), for a minimum clock frequency of 166·7 MHz. If, however, $M=25$ is chosen instead of 50, using (12), then, from (13), $83{\cdot}3\,\text{MHz} \leqslant f_{clk} \leqslant 100\,\text{MHz}$ in order for this system to latch the correct data (since $T_{cp}=T_e+4T_{fN}=10$ ns) and satisfy the maximum latency constraint.

4.2. *Required clock frequency*

In applications where the maximum clock frequency is specified and $f_{clk_{MAX}}$ constrains the design space, the latency and the number of registers can be determined from (4) and (14), respectively, and

$$M=\frac{NT_{fN}}{T_{cp}-T_{eM}} \tag{14}$$

Thus, as shown in Fig. 3, for a given maximum or required clock frequency and knowledge of the average logic, register, and clock delay characteristics of a global data path, the minimum latency and the required level of pipelining can be determined directly. An example is provided below describing this clock frequency constrained design problem.

Example 2: *Determining latency for a specified clock frequency*

This example describes how the latency is determined for a specified clock frequency (or maximum clock frequency). This example can be explained in the context of Fig. 3 where in this case the maximum clock frequency, not the maximum latency, constrains the design space. The appropriate level of pipelining to minimize L while satisfying $f_{clk_{MAX}}$ is determined by the intersection of the clock period and the $f_{clk_{MAX}}$ curves, defining both M and L.

Equations (4) and (14) can be used to determine the specific latency and the number of registers, respectively, for a given maximum clock frequency. Thus, assume a 100 stage data path is analysed with an average stage delay T_{fN} of 2 ns and an average register and skew delay T_{eM} of 5 ns. L and M can be directly determined for a specified clock frequency as shown in Fig. 7. If the maximum clock frequency is 40 MHz (the clock period is 25 ns), then the number of local data paths (M), given by (14), is 10 (requiring 11 pipeline registers). These ten local data paths add 50 ns to the latency of the global data path, defining a total L of 250 ns.

4.3. *Unconstrained design requirement*

Every register added to a data path increases the latency of a system by the added time required to move the data signal in and out of the register, T_{REG}. This is typically accepted in order to increase the system clock frequency (Cotton 1965, Cappello and Steiglitz 1982, 1983, Leiserson and Saxe 1981, Cappello *et al.* 1984, Jump and Ahuja 1978, Hatamian *et al.* 1987). However, as additional stages of pipelining are inserted into a data path, the marginal utility of the increased clock

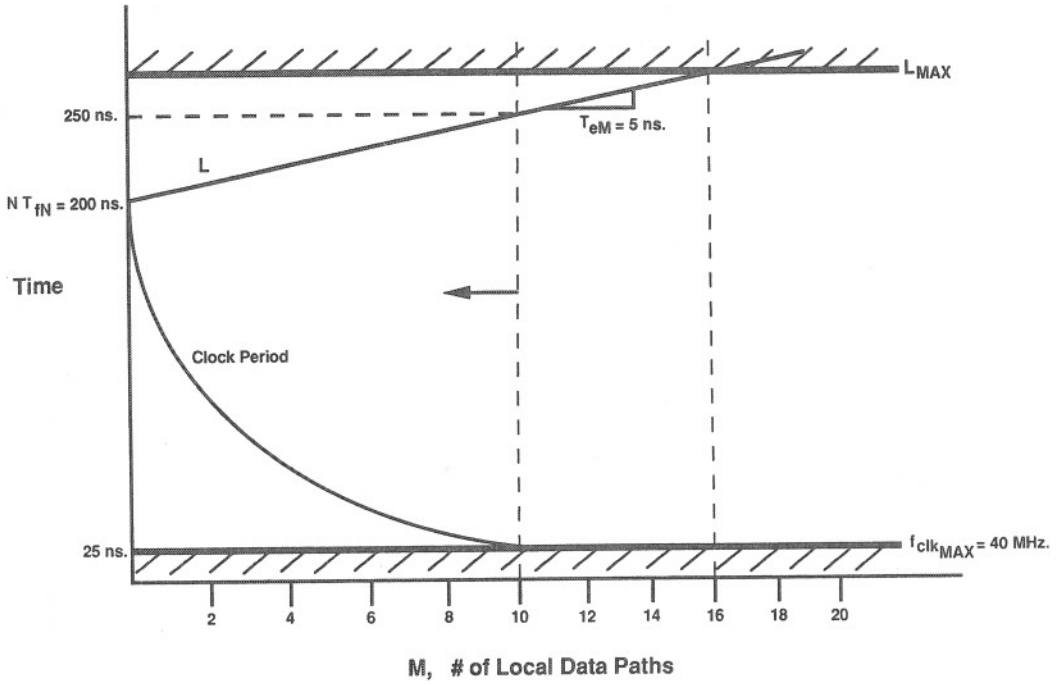

Figure 7. Example of design paradigm with constraining clock frequency.

rate is approached. This occurs when the time required to move data in and out of the register becomes comparable to or greater than the time incurred performing the logic functions.

Each additional register increases L by T_e and decreases the maximum clock period by the decreased logic delay of the critical path. There exists an application-specific level of pipelining M where the increase in latency costs the system more than the increase in clock frequency benefits the system. In order to quantify this, an arbitrary performance criterion (the pipelining efficiency, P_e) is defined to describe the performance cost of latency. P_e is a measure of the relative performance penalty incurred for n stages of logic per one pipelined local data path and describes the cost in performance (in increased latency) incurred by the insertion of a single additional pipeline register. This normalized function, shown in (15), is the ratio of the total local logic delay to the total local data path delay, thereby defining what percentage of the local data path delay is logic related and what percentage is register related. As n increases, the ratio of the total local logic delay to the total local data path delay increases toward unity, reaching it when n is infinite (or practically, when the total local logic delay is much greater than the register delay).

$$P_e = \frac{\sum_{i=1}^{n} T_{fi}}{T_{PD}} = \frac{\sum_{i=1}^{n} T_{fi}}{\sum_{i=1}^{n} T_{fi} + T_{REG}} \tag{15}$$

The benefit of inserting a register into a data path is increased clock frequency. If one starts with (2), a new relationship for clock frequency can be expressed as

$$f_{clk} \leqslant \frac{1}{T_{REG} + \sum_{i=1}^{n} T_{fi} + T_{SKEW}} = \frac{1}{\sum_{i=1}^{n} T_{fi} + T_e} \tag{16}$$

Thus, a measure of the cost/benefit of inserting registers into an N stage data path is the function $P_e f_{clk}$, where P_e increases for increasing n and f_{clk} decreases for increasing n. $P_e f_{clk}$ is thus a figure of merit for representing the performance advantages and disadvantages of pipelining. A different cost/benefit function could be applied if the effects of increased area, for example, were also of significant importance (Leiserson and Saxe 1981, Cappello *et al.* 1984, Siomalas and Bowen 1985). Cappello *et al.* (1984) use an AP-product as a figure of merit (A is the chip area and P is the clock period) to optimize the speed/area performance of a pipelined system. The results described here, however, emphasize optimal latency and clock frequency over area/speed optimization. These results, coupled with the approaches described in §5 for analysing the effects of other performance parameters on the design space, provide a systematic methodology for designing those systems in which both latency and clock frequency are of primary importance and other key parameters, such as area and power dissipation, are of secondary importance.

If (15) is combined with (16), the figure of merit $P_e f_{clk}$ can be expressed as

$$P_e f_{clk} = \frac{nT_{fN}}{(nT_{fN} + T_{REG})} \cdot \frac{1}{(T_{REG} + nT_{fN} + T_{SKEW})} \tag{17}$$

An optimal number of logic stages between pipeline registers N_{opt} is obtained from (17) and (18) by determining where the product $P_e f_{clk}$ is maximized or where

$$\frac{d(P_e f_{clk})}{dn} = 0 \tag{18}$$

From (18), N_{opt}, the optimal number of logic stages per local data path for a high speed synchronous digital system, is obtained as

$$N_{opt} = \frac{1}{T_{fN}} (T_{REG}(T_{REG} + T_{SKEW}))^{1/2} \tag{19}$$

Under the condition of an ideal clock distribution network with zero clock skew, (19) simplifies to (20):

$$N_{opt} = \frac{T_{REG}}{T_{fN}} \tag{20}$$

N_{opt}, in (20), is the ratio of the register delay overhead to the average logic and interconnect stage delay of the global data path. If T_{REG} is much less than T_{fN}, which occurs when T_{fN} represents the delay of a large multi-level function, the cost of inserting registers is small and N_{opt}, from (20), should be as small as feasible (since N_{opt} must be an integer, its smallest realizable value is one stage) or one should pipeline as often as the system permits. If T_{REG} is much greater than T_{fN}, which often exists when operating at the level of individual logic stages, then the

cost of inserting registers is high and N_{opt} is some large number determined from (20). Another interpretation of (20) is that the optimal number of logic stages between registers occurs when the total logic path delay NT_{fN} equals the total register delay T_{REG}, thereby maximizing $P_e f_{clk}$.

T_{SKEW} in (19) can be positive, negative, or zero with the constraint that if T_{SKEW} is negative, its magnitude must be less than T_{REG}. It is interesting to note that the effect of clock skew on N_{opt} is dependent upon its relative magnitude with respect to T_{REG} and T_{fN}. Thus, if T_{REG} is large with respect to T_{SKEW}, the relation essentially reduces to (20). Also, positive clock skew adds directly to T_{REG} and increases the cost of pipelining which increases the recommended number of logic stages between registers and quantifies how the clock distribution network affects the optimal design of a high speed data path.

The performance of a pipelined synchronous system can be maximized when the cost of the registers is minimal, thereby permitting the frequent insertion of pipeline registers and a higher overall system clock rate. As implied by (19) and shown in Fig. 5, if T_{SKEW} is negative, the overall performance cost of pipelining decreases.

The maximum permissible negative clock skew in (19) can be represented by T_e, where T_e is the margin of error or the acceptable tolerance as defined by (5). Equation (19) can then be rewritten as (21):

$$N_{opt} = \frac{(T_{REG} T_e)^{1/2}}{T_{fN}} \tag{21}$$

For a given N_{opt}, T_e, and average gate delay T_{fN}, the optimal frequency at which a particular data path (and system) should operate and the optimal level of pipelining can be computed from (22) and (23), respectively. In an actual application, N_{opt} should be rounded to an integer value, hereby denoted as $\bar{N}_{opt}$. Rounding to the next highest (lowest) integer value improves (degrades) the density since fewer (more) registers are used but the clock frequency decreases (increases) since there are more (fewer) logic stages per local data path:

$$f_{clk_{opt}} \leqslant \frac{1}{T_{REG} + \bar{N}_{opt} T_{fN} + T_{SKEW}}$$

$$= \frac{1}{\bar{N}_{opt} T_{fN} + T_e} \tag{22}$$

$$M_{opt} = \frac{N}{\bar{N}_{opt}} \tag{23}$$

An example is provided below describing this unconstrained design problem.

Example 3: *Determining an optimal trade-off between clock frequency and latency*

This example describes a design problem in which neither the clock frequency nor the latency is constrained by the circuit specifications. This example can be explained in the context of Fig. 3 where the target latency and clock frequency are less than L_{max} and $f_{clk_{req}}$, respectively, and the design space is unconstrained. Therefore, a trade-off between L and f_{clk} must be made.

Equation (19) can be used to determine the appropriate number of logic stages between registers. Thus, assume a 30 stage data path is under consideration with an average stage delay T_{fN} of 4 ns, an average (positive) clock skew of 4 ns, and a register delay of 6 ns. Equation (19) applies and N_{opt} is obtained as

$$N_{opt} = \frac{1}{T_{fN}} (T_{REG}(T_{REG} + T_{SKEW}))^{1/2}$$

$$N_{opt} = \tfrac{1}{4}(6(6+4))^{1/2} = 1{\cdot}94$$

Therefore, since N_{opt} is 1·94, there should be two logic stages per pipelined data path. From (22), the optimal maximum clock frequency is 55·6 MHz. Since $N_{opt} = 2$ and $N = 30$, M_{opt}, from (23), is 15 local data paths (or 16 pipeline registers). Since T_{eM} $(T_{REG} + T_{SKEW})$ is 10 ns, the total latency of this system, from (4), is 270 ns. Thus, an optimal implementation of this high performance global data path is 15 local data paths operating at a maximum of 55·6 MHz with a latency of 270 ns.

5. Incorporation of additional performance criteria

In this paper three types of design problems have been described, where the goal is either to provide the highest clock frequency without exceeding a maximum latency, to realize a required clock frequency while minimizing the latency, or to provide a reasonable trade-off between the latency and the clock frequency by applying the concept of marginal utility to both of these parameters. All of these strategies permit the incorporation of additional performance parameters and these additional degrees of freedom can be described in the context of the design paradigm described in § 3. A powerful approach for dealing with these design problems is the use of the pipeline factor M to drive the design process to reduce area or power consumption or both. This design approach is represented in Fig. 3 by a choice of M less than that constrained by L_{max} or $f_{clk_{MAX}}$. With this methodology, the effects of other performance goals on the design space can be explored while still satisfying an application specific L and f_{clk}.

Different options are possible to improve the speed/area/power implementation of a pipelined synchronous system. These include

(*a*) pipelining less frequently;

(*b*) changing the nature of the logic stages;

(*c*) changing the system architecture; and

(*d*) a combination of the aforementioned design options.

For example, pipelining less often saves register area and power dissipation. Alternatively, one can change the nature of the logic stages by either using smaller logic stages which consume less circuit area and less source and sink current, thereby dissipating less power, or by selecting a less aggressive high speed technology which could save area and power dissipation and possibly save cost through increased yield. An example of the latter is using a standard silicon-based technology such as CMOS instead of higher speed and more expensive technologies such as GaAs and high electron mobility transistors (HEMT). Another option is to implement the system with a more power- and area-efficient lower-speed

architecture which still satisfies system specifications. Finally, by combining each of these design options, a highly desirable system implementation could result.

5.1. *Pipelining less frequently*

Pipelining is applied to increase system clock frequency. For those applications which require lower levels of clock frequency (i.e. the problem is unconstrained or the maximum latency constrains the design problem), a choice of M can be made which is less than that defined by L_{max} or $f_{clk_{MAX}}$. If fewer registers are used since M is smaller, the total system area and power dissipation is decreased. For example, for a 100 stage global data path composed of 20 local data paths, if M is decreased by 20%, the total area and power dissipation for the complete system would decrease by approximately 7·5% and 4·5%, respectively. Thus, as M is decreased, the design space becomes constrained by the application specific L_{max} and $f_{clk_{MAX}}$. Since less area and power dissipation are required in this new system implementation, this new choice of M represents an improvement as compared to the unconstrained design problem.

As M is decreased, the latency and clock period change (see Fig. 3). L decreases and T_{cp} increases (the clock frequency decreases) for decreasing M, as shown in (4) and (9). T_{cp} increases since n, the number of logic stages between registers, increases for decreasing M (as shown by (6) and (7)). Thus, the level of pipelining M can be decreased until either L equals L_{max} or T_{cp} equals $T_{cp_{req}}$, where $T_{cp_{req}}$ is the maximum required clock period (minimum required clock frequency). This constrains the design space and defines the appropriate level of pipelining. Design equations describing these conditions are

$$M_1 = \frac{L_{max} - NT_{fN}}{T_{eM}} \tag{24}$$

$$M_2 = \frac{NT_{fN}}{T_{cp_{req}} - T_{eM}} \tag{25}$$

where the appropriate level of pipelining M is defined by

$$M = \text{Max}\{M_1, M_2\} \tag{26}$$

Thus, for unconstrained design problems, the level of pipelining can be decreased, saving circuit area and power dissipation, until either the application-defined maximum latency or the required clock frequency is reached. Once either constraining design condition, (24) or (25), is satisfied, the limiting application-specific level of pipelining M is defined.

5.2. *Changing the nature of the logic stages*

If the problem is not constrained by the maximum latency or the required clock frequency, another approach for improving the speed/area/power dissipation of a particular system implementation is, for a given logic architecture (i.e. N is constant), to change the nature of the logic stages used. In an unconstrained design problem, different approaches can be used to optimize the delay of the logic stages. Either a less aggressive technology can be used or the logic stages can be designed to be slower (since T_{fN} can be larger), permitting the circuit area, for a given technology, to be smaller. These smaller logic stages source less current and

therefore dissipate less power. The design paradigm graphically expresses the significance of this approach. Since T_{fN} increases, from (4) and (9), the latency and clock period also both increase. In Fig. 3, both the latency curve and the clock period curve shift upward for increasing T_{fN}. Thus, T_{fN} can be increased until either L equals L_{max} or T_{cp} equals $T_{cp_{req}}$, thereby constraining the design space and defining the correct level of average delay per logic stage. Design equations describing these conditions are

$$T_{fN_1} = \frac{L_{max} - MT_{eM}}{N} \tag{27}$$

$$T_{fN_2} = \frac{M}{N}(T_{cp_{req}} - T_{eM}) \tag{28}$$

where the appropriate stage delay T_{fN} is defined by

$$T_{fN} = \text{Min}\{T_{fN_1}, T_{fN_2}\} \tag{29}$$

Thus, for unconstrained design problems, a slower technology can be used or the logic stages can be designed to minimize area and power dissipation instead of minimizing delay. In either case, once a target logic stage delay is known, many approaches exist for making design trade-offs among speed, area, and power dissipation for a variety of different technologies. Some examples of these approaches, in which the logic delay is described in terms of specific technological, geometric, and electrical characteristics, are provided by Lin and Linholm (1975), Jaeger (1975), Kanuma (1983), Lee and Soukup (1984), Auvergne *et al.* (1987), and Huss and Gerbershagen (1987).

In either approach, whether a different technology is used or the circuit design of the logic stages emphasizes area and power over speed (or a combination of the two), the new system implementation has advantages for the particular application. While still satisfying the application-specific latency and clock frequency requirements, less area and power are required.

5.3 *Changing the system architecture*

Other possible approaches besides decreasing M or increasing T_{fN} can be considered for a given unconstrained system. For example, new system architectures can be considered which are slower but consume less area and power. This equates to decreasing N, the number of logic stages per global data path, but increasing T_{fN}. A simple example of this approach is replacing a high speed carry look-ahead adder with a serial adder, assuming the added performance of the carry look-ahead adder is not required. In this instance, the serial adder would be much slower than the carry look-ahead adder but consume considerably less area and power. In the context of the design paradigm depicted in Fig. 3 and described in § 3, N would decrease since more of the function is done serially, but T_{fN} would increase since the total delay from system input to system output is greater, thereby increasing the average logic stage delay. This approach can be used until the increase in T_{fN} is significantly greater than the decrease in N, shifting both curves in Fig. 3 upward until either the L_{max} or $f_{clk_{req}}$ constraint is reached.

Any subset of these aforementioned approaches can be combined during the design phase in order to reach a more desirable system implementation. Thus, the

design paradigm, coupled with design principles discussed in this section, can be used with application-specific performance constraints to explore the effects on system requirements of other parameters, such as area and power dissipation.

6. Conclusions

Clock frequency and latency are convenient parameters for describing the performance of certain high speed synchronous digital systems. The results of this paper deal directly with the systematic design of those systems based upon these two performance attributes. Additional performance attributes, such as power dissipation and area, are also addressed and design methodologies are described for those applications where the system requirements are not limited by the available design space.

In feed-forward non-recursive systems, global data paths are often partitioned into local pipelined data paths, thereby decreasing the delay of the critical paths and increasing the clock frequency, albeit with an increase in latency. The graphical design paradigm presented herein permits one to analyse how the performance of a synchronous system is affected by its degree of pipelining. This perspective permits the development of design equations for describing pipelined data paths in terms of the logic, interconnect, and register delays, clock skew, the performance efficiency of pipelining, and the total number of logic stages per local data path.

The results described in this paper specifically discuss three types of design problems; namely, that in which

(*a*) $L_{\max}$ constrains the design space;

(*b*) $f_{\text{clk}_{MAX}}$ constrains the design space; and

(*c*) the design space is unconstrained and a trade-off must be made between L and f_{clk}.

Design equations have been presented which permit a solution to be determined for each type of problem. The solution suggested for the unconstrained design problem is the use of an algorithm which trades off the effects of increased latency and increased clock frequency for increasing levels of pipelining.

There is an important class of practical system applications, such as radar, sonar, and high speed computing, which requires both high clock frequency and minimal latency and for which there is a need for developing a design methodology which satisfies their application-specific performance objectives. The results presented in this paper describe a systematic strategy for designing high performance systems in which both clock frequency and latency are of primary interest.

ACKNOWLEDGMENT

This research was supported in part by a Hughes Aircraft Doctoral Fellowship.

Appendix

Delay components of a local data path

Since logic paths are composed of only logic stages and interconnect sections,

the total delay through a logic path can be modelled as the sum of the delay through the individual logic stages and interconnect sections. In order to represent the delay through the system, each individual RC interconnect section and logic stage is combined as a single delay component of the logic path. This permits one to define the time required for the data signal to propagate through the ith distributed RC interconnect section T_i and logic stage L_i as T_{fi} and the average delay of all the logic and interconnect stages per data path as T_{fN}.

Each added register generates delay components which are added to the logic and interconnect delays and which, when summed with the clock skew, must be less than the clock period. Since there are fewer logic stages in a pipelined data path than an unpipelined data path, the clock rate is higher in a pipelined system. The register related delay components which are added to the path delay T_{PD}, as observed in Fig. 1, originate in R_i and R_f. T_{c-Q} is the time interval between the arrival of the clock signal at R_i and the appearance of the data signal at the register output. The time required for the signal at the output of the final logic stage to propagate through the $(N+1)$th interconnect section and latch into the final register R_f is the set-up time $T_{set\text{-}up}$.

Thus, for an N stage logic path, the delay through a local data path can be expressed as

$$T_{PD} = T_{c-Q} + \sum_{i=1}^{N} T_{fi} + T_{set\text{-}up} \tag{A 1}$$

Equation (A 1) is composed of the delay required to get out of and into the initial and final registers, respectively, and the time required to propagate through N stages of logic and $N+1$ sections of interconnect. If T_{REG} represents the total register related delay of both R_i and R_f, then

$$T_{REG} = T_{c-Q} + T_{set\text{-}up} \tag{A 2}$$

and equation (A 1) can be written as

$$T_{PD} = T_{REG} + \sum_{i=1}^{N} T_{fi} \tag{A 3}$$

Thus, the total time to move a data signal through a data path is composed of the overhead requirements to get in and out of the register as well as the time required to perform the logical operations.

References

Auvergne, D., Deschacht, D., and Robert, M., 1987, Explicit formulation of delays in CMOS VLSI. *Electronic Letters*, **23**, 741–742.

Cappello, P. R., LaPaugh, A., and Steiglitz, K., 1984, Optimal choice of intermediate latching to maximize throughput in VLSI circuits. *I.E.E.E. Transactions on Acoustics, Speech, and Signal Processing*, **32**, 28–33.

Cappello, P. R., and Steiglitz, K., 1982, Bit-level fixed-flow architectures for signal processing. *Proceedings of the I.E.E.E. International Conference on Circuits and Computers*, pp. 570–573; 1983, Completely-pipelined architectures for digital signal processing. *I.E.E.E. Transactions on Acoustics, Speech, and Signal Processing*, **31**, 1016–1023.

Cotton, L. W., 1965, Circuit implementation of high-speed pipeline systems. *Proceedings of the Fall Joint Computer Conference*, pp. 489–504.

FRIEDMAN, E. G., and MULLIGAN, J. H. JR, 1991, Clock frequency and latency in synchronous digitial systems. *I.E.E.E. Transactions on Acoustics, Speech, and Signal Processing*, **39**.

FRIEDMAN, E. G., and POWELL, S., 1986, Design and analysis of a hierarchical clock distribution system for synchronous standard cell/macrocell VLSI. *I.E.E.E. Journal of Solid-State Circuits*, **21**, 240–246.

HATAMIAN, M., 1988, Understanding clock skew in synchronous systems. *Concurrent Computations (Algorithms, Architecture, and Technology)*, edited by S. K. Tewksbury, B. W. Dickinson, and S. C. Schwartz (New York: Plenum Publishing), Chap. 6.

HATAMIAN, M., and CASH, G. L., 1987, Parallel bit-level pipelined VLSI designs for high speed signal processing. *Proceedings of the Institute of Electrical and Electronics Engineers*, **75**, 1192–1202.

HATAMIAN, M., HORNAK, L. A., LITTLE, T. E., TEWKSBURY, S. T., and FRANZON, P., 1987, Fundamental interconnection issues. *AT&T Technical Journal*, **66**, 13–30.

HUSS, S. A., and GERBERSHAGEN, M., 1987, Signal delay calculation for integrated CMOS circuits. *AEU/Archiv für Elektronik und Ubertragungstechnik*, **41**, 214–222.

JAEGER, R. C., 1975, Comments on 'An optimized output stage for MOS integrated circuits'. *I.E.E.E. Journal of Solid-State Circuits*, **10**, 185–186.

JUMP, J. R., and AHUJA, S. R., 1978, Effective pipelining of digital systems. *I.E.E.E. Transactions on Computers*, **27**, 855–865.

KANUMA, A., 1983, CMOS circuit optimization. *Solid-State Electronics*, **26**, 47–58.

LEE, C., and SOUKUP, H., 1984, An Algorithm for CMOS timing and area optimization. *I.E.E.E. Journal of Solid-State Circuits*, **19**, 781–787.

LEISERSON, C. E., and SAXE, J. B., 1981, Optimizing synchronous systems. *Proceedings of 22nd Annual Symposium on the Foundations of Computer Science*, pp. 23–26.

LIN, H. C., and LINHOLM, L. W., 1975, An optimized output stage for MOS integrated circuits. *I.E.E.E. Journal of Solid-State Circuits*, **10**, 106–109.

SJOMALAS, K. O., and BOWEN, B. A., 1985, Synthesis of efficient pipelined architectures for implementing DSP operations. *I.E.E.E. Transactions on Acoustics, Speech, and Signal Processing*, **33**, 1499–1508.