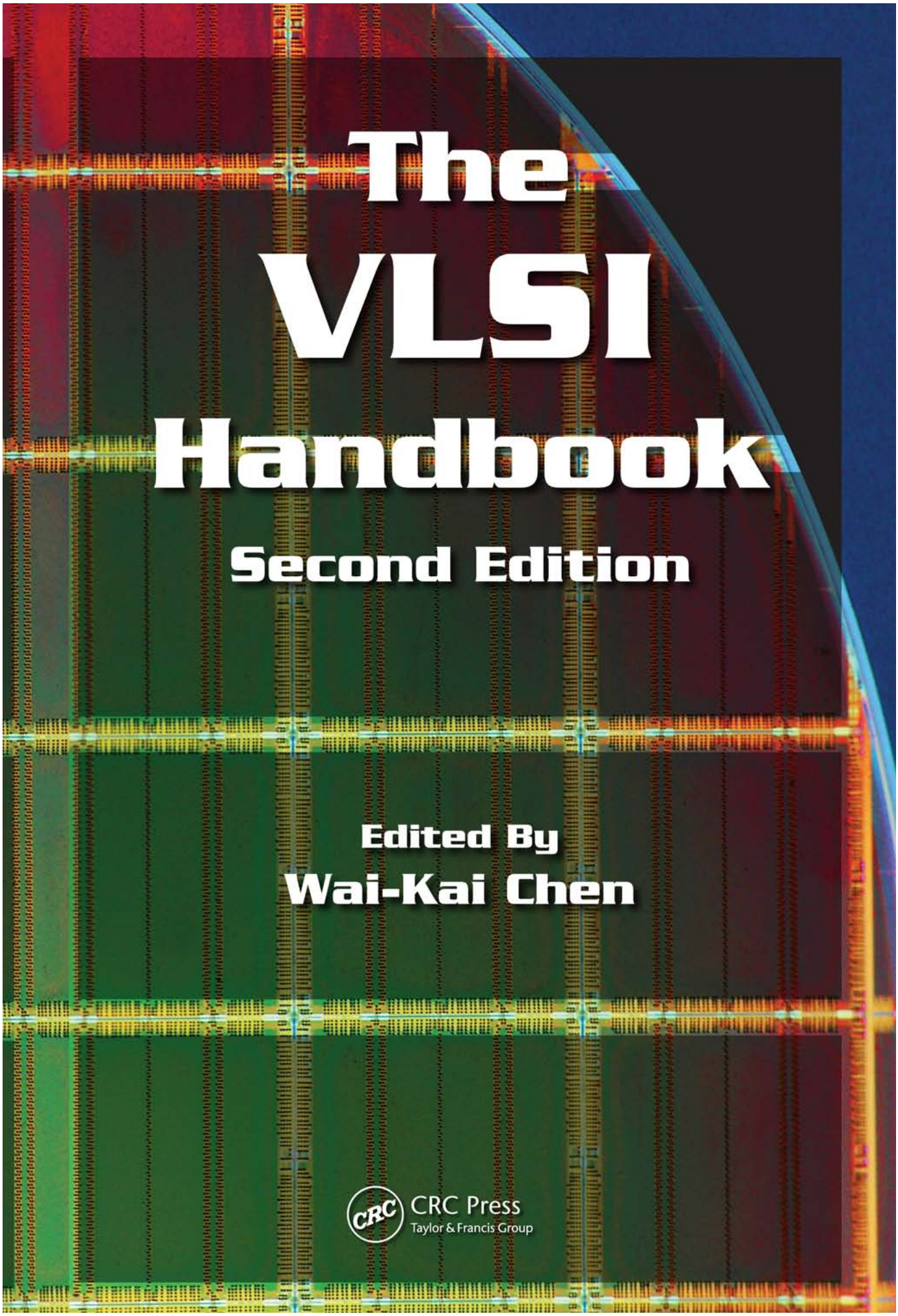

The
VLSI
Handbook
Second Edition
Edited By
Wai-Kai Chen
CRC
CRC Press
Taylor & Francis Group

50

System Timing

Baris Taskin
Drexel University

Ivan S. Kourtev
University of Pittsburgh

Eby G. Friedman
University of Rochester

CONTENTS

50.1 Introduction

The concept of *data* or *information* processing arises in a variety of fields. Understanding the principles behind this concept is fundamental to computer design, communications, manufacturing process control, biomedical engineering, and an increasingly large number of other areas of technology and science. It is impossible to imagine modern life without computers for generating, analyzing, and retrieving large amounts of information, as well as for communicating information to end users regardless of their location.

Technologies for designing and building microelectronics-based computational equipment have been steadily advancing ever since the first commercial *discrete integrated circuits* were introduced in the late 1950s [1].* As predicted by *Moore's law* in the 1960s [2], integrated circuit (IC) densities have been doubling

*Monolithic ICs were introduced in the 1960s.

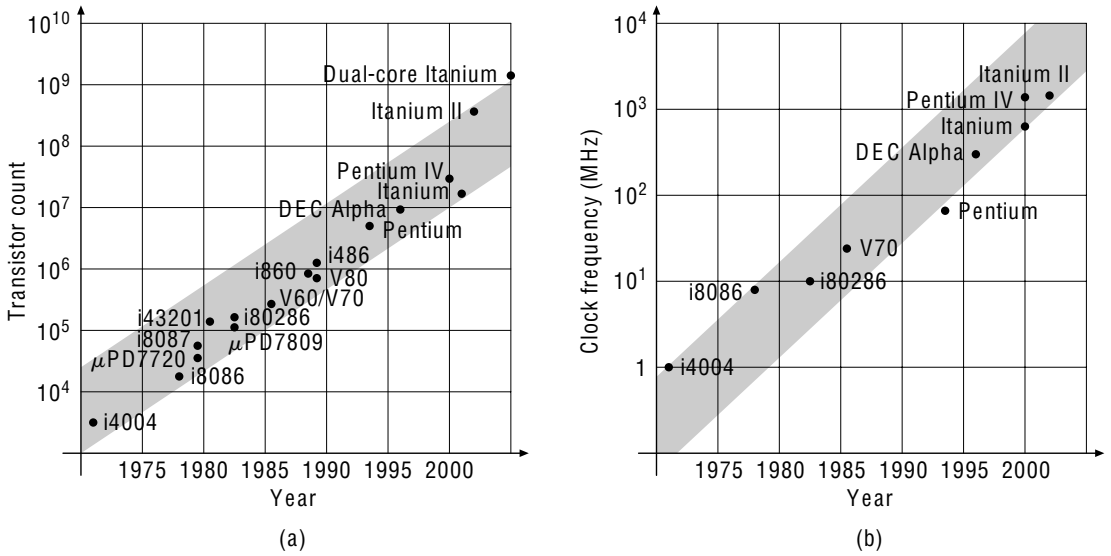


FIGURE 50.1 Moore's law—exponential increase in circuit integration and clock frequency. (a) Evolution of the number of transistors per IC. (b) Evolution of clock frequency.

approximately every 18 months, and this doubling in size has been accompanied by a similar exponential increase in circuit speed (or more precisely, clock frequency). These trends of steadily increasing circuit size and clock frequency are illustrated in Figure 50.1(a) and Figure 50.1(b), respectively. As a result of this revolution in semiconductor technology, it is not unusual for modern integrated circuits to contain hundreds of millions of switching elements (i.e., transistors) packed into a chip area as large as 500 mm^2 [3–6]. Such technological capability is due to advances in both design methodologies and physical manufacturing technologies. Research and experience demonstrate that this trend of exponentially increasing integrated circuit-based computational power will continue into the foreseeable future.

Integrated circuit performance is typically characterized [7] by the *speed of operation,* the available *circuit functionality*, and the *power consumption*, and there are multiple factors which directly affect these performance characteristics. While each of these factors is significant, on the technological side, increased circuit performance has been largely achieved by the following approaches:

- Reduction in feature size (technology scaling), that is, the capability of manufacturing physically smaller and faster device structures
- Increase in chip area, permitting a larger number of circuits and therefore greater on-chip functionality
- Advances in packaging technology, permitting the increasing volume of data traffic between an integrated circuit and its environment as well as the efficient removal of heat generated during circuit operation

The most complex integrated circuits are referred to as very large scale integration (VLSI) circuits. This term describes the complexity of modern integrated circuits consisting of hundreds of thousands to many millions of active transistor elements. Presently, the leading integrated circuit manufacturers have a technological capability for the mass production of VLSI circuits with feature sizes as small as 65 nm [8]. These technologies are identified with the terms *nanometer* or *very deep submicrometer* (VDSM) technologies.

As these dramatic advances in fabrication technologies take place, integrated circuit performance is often limited by effects closely related to the very reasons behind these advances such as small geometry interconnect structures. Circuit performance becomes strongly dependent and limited by electrical issues that are particularly significant in deep submicrometer integrated circuits. *Signal delay* and related

waveform effects are among those phenomena that have great impact on high performance integrated circuit design methodologies and the resulting system implementation. In the case of fully synchronous VLSI systems, these effects have the potential to create catastrophic failures due to the limited time available for signal propagation between logic gates.

Specifically, in Section 50.2, general timing and operational properties of synchronous circuits are presented. In Section 50.3, the modeling of synchronous circuit components (suitable for computer manipulation) is presented. Also in Section 50.3, the impact of the clock distribution network on circuit timing is described. In Section 50.4, system timing is analyzed. First, system timing properties of edge-triggered and level-sensitive circuits are analyzed. Then, clock skew scheduling methodologies for both types of circuit structures are described. Last, the limitations to improvements in circuit performance achievable through clock skew scheduling are presented. The section is finalized with an appendix containing a glossary of the many terms used throughout this chapter.

50.2 Synchronous VLSI Systems

Owing to the relative simplicity in the design process, the analysis and optimization of VLSI circuits are generally based on logic components operating under a fully synchronous synchronization scheme. In the following sections, these design concepts are briefly reviewed and related fundamental properties are identified. The operational components of VLSI systems relevant to system timing are highlighted.

50.2.1 General Overview

Typically, a digital VLSI system performs a complex computational algorithm, such as a fast Fourier transform or a RISC[†] architecture microprocessor. Although modern VLSI systems contain a large number of components, these systems normally employ only a limited number of different kinds of *logic elements* or *logic gates*. Each logic element accepts certain input signals and computes an output signal for use by other logic elements. At the logic level of abstraction, a VLSI system is a *network* of hundreds of thousands or more logic gates whose terminals are *interconnected* by wires to implement a target algorithm.

The switching variables acting as inputs and outputs of a logic gate in a VLSI system are represented by tangible physical quantities,[‡] while a number of these devices are interconnected to yield the desired function of each logic gate. The specific physical characteristics are collectively summarized with the term *technology*, encompassing details such as the type and behavior of the devices that can be built, the number and sequence of manufacturing steps, and the impedance of the different interconnect materials. Today, several technologies make possible the implementation of high-performance VLSI systems—these technologies are best exemplified by CMOS, bipolar, BiCMOS, and gallium arsenide [9,10]. CMOS technology, in particular, exhibits many desirable performance characteristics, such as low power consumption, high density, ease of design, and moderate to high speed. Owing to these excellent performance characteristics, CMOS technology has become the dominant VLSI technology used today.

The design of a digital VLSI system requires a great deal of effort to consider a broad range of architectural and logical issues; that is, choosing the appropriate gates and interconnections among these gates to achieve the required circuit function. No design is complete, however, without considering the *dynamic* (or transient) characteristics of the signal propagation, or, alternatively, the changing behavior of signals with *time*. Every computation performed by a switching circuit involves multiple signal transitions between logic states and requires a *finite* amount of time to complete. The voltage at every circuit node must reach a specific value for the computation to be completed. State-of-the-art integrated circuit design is therefore largely centered around the difficult task of predicting and properly interpreting signal waveform shapes at various points in a circuit.

[†]RISC, reduced instruction set computer.

[‡]Quantities such as the *electrical voltages* and *currents* in electronic devices.

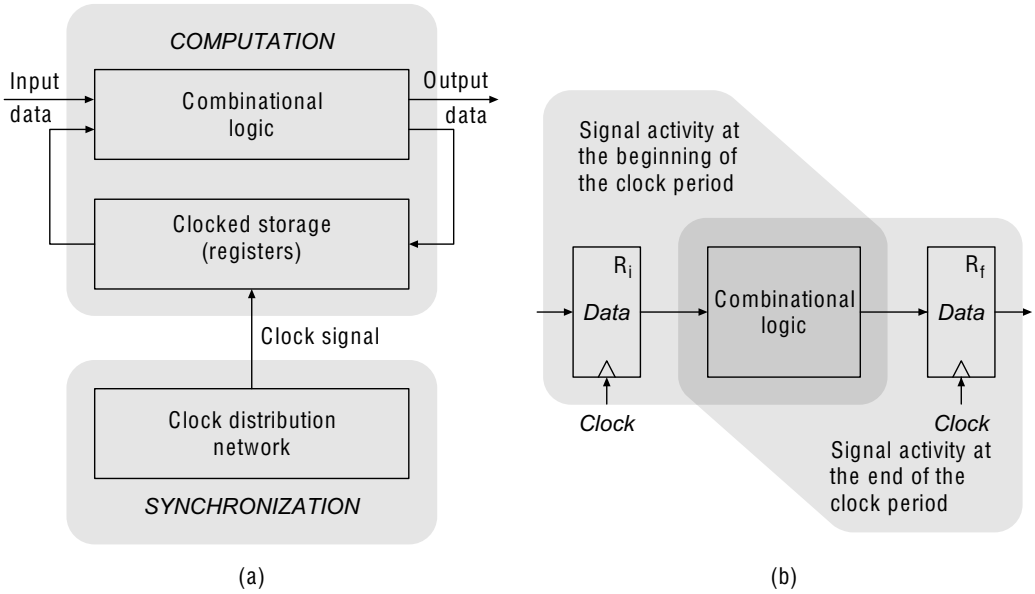


FIGURE 50.2 A synchronous system. (a) Finite-state machine model of a synchronous system. (b) A local data path.

In a typical VLSI system, millions of signal transitions determine the individual gate delays and the overall speed of the system. Some of these signal transitions can be executed *concurrently* while others must be executed in a strict *sequential* order [11]. The sequential occurrence of the latter operations—or signal transition *events*—must be properly coordinated in time such that logically correct system operation is guaranteed and the results are reliable (in the sense that these results can be repeated). This coordination is known as *synchronization* and is critical to ensuring that any pair of logical operations in a circuit with a precedence relationship proceed in the proper order. In modern digital integrated circuits, synchronization is achieved at all stages of the system design process and operation by a variety of techniques, known as a *timing discipline* or *timing scheme* [9,12–14]. With some exceptions, these circuits are based on a *fully synchronous* timing scheme, specifically developed to cope with the finite speed required by the physical signals to propagate through the system.

An example of a *fully synchronous* system is shown in Figure 50.2(a). As illustrated in Figure 50.2(a), there are three recognizable components in this system. The first component—the logic gates, collectively referred to as the *combinational logic*—provides the range of operations that a system executes. The second component—the *clocked storage* elements or simply the *registers*—are elements that store the results of the logical operations. Together, the combinational logic and registers constitute the *computational* portion of the synchronous system and are interconnected in a way that implements the required system function. The third component of the synchronous system—known as the *clock distribution network*—is a highly specialized circuit structure which does not perform a computational process but rather provides an important control capability. The clock generation and distribution network controls the overall synchronization of the circuit by *generating* a time reference and properly *distributes* this time reference to every register.

The normal operation of a synchronous system, such as the example finite-state machine shown in Figure 50.2(a), consists of the iterative execution of computations in the combinational logic followed by the storage of the processed results in the registers. The actual process of storage is temporally controlled by the clock signal and occurs once the signal transients in the logic gate outputs are completed and the outputs have settled to a valid state. At the beginning of each computational cycle, the inputs of the system together with the data stored in the registers initiate a new switching process. As time proceeds, the signals propagate through the logic, generating results at the logic output. By the end of the clock period, these results are stored in the registers. During the following clock cycle, the stored data values start propagating through the logic, progressing toward the system outputs.

The operation of a digital system can therefore be thought of as the sequential execution of a large set of simple computations that occur concurrently in the combinational logic portion of the system. The concept of a *local data path* is a useful abstraction for each of these simple operations and is shown in Figure 50.2(b). The magnitude of the delay of the combinational logic is bound by the requirement of storing data in a register within a clock period. The initial register R_i is the storage element at the beginning of the local data path and provides some or all of the input signals for the combinational logic at the beginning of the computational cycle (defined by the beginning of the clock period). The *combinational path* ends with the data successfully latching within the final register R_f where the results are stored at the end of the computational cycle. Registers act as *sources* and *sinks* for the data between the clock cycles.

50.2.2 Advantages and Drawbacks of Synchronous Systems

The behavior of a fully synchronous system is well defined and controllable as long as the *time window* provided by the clock period is sufficiently long to allow every signal in the circuit to propagate through the required logic gates and interconnect wires and successfully latch within the final register. In designing the system and choosing the proper clock period, however, two contradictory requirements must be satisfied. First, the smaller the clock period, the more computational cycles can be performed by the circuit in a given amount of time. Alternatively, the time window defined by the clock period must be sufficiently long such that the slowest signals reach the destination registers before the current clock cycle is concluded and the following clock cycle is initiated.

Such an organization of computation has certain clear advantages that propel a fully synchronous timing scheme to remain as the primary choice for digital VLSI systems:

- The properties and variations are simple and well understood.
- The scheme eliminates the nondeterministic behavior of the propagation delay in the combinational logic (due to environmental and process fluctuations and unknown input signal patterns) such that the system exhibits a deterministic behavior corresponding to the implemented algorithm.
- The circuit design does *not* need to be concerned with glitches in the combinational logic outputs, so the only relevant dynamic characteristic of the logic is the *propagation delay.*
- The state of the system is completely defined within the storage elements—this fact greatly simplifies certain aspects of the design, debug, and test phases in developing a large system.

A synchronous paradigm, however, also has certain limitations that make the design of synchronous VLSI systems increasingly challenging:

- This synchronous approach has a serious drawback in that the timing scheme requires the overall circuit to operate as slow as the *slowest* register-to-register path. Thus, the global speed of a fully synchronous system depends upon those paths in the combinational logic with the largest delays—these paths are also known as the *worst-case* or *critical* paths. In a typical VLSI system, the propagation delays in the combinational paths are distributed unevenly so there may be many paths with delays much smaller than the clock period. Although these paths could operate correctly at a lower clock period—higher clock frequency—it is those paths with the largest delays that bound the clock period, thereby imposing a limit on the overall system speed. This imbalance in propagation delays is sometimes so dramatic that the system speed is dictated by only a handful of very slow paths.
- The clock signal has to be distributed to tens of thousands of storage registers scattered throughout the system. A significant portion of the system area and dissipated power is therefore devoted to the clock distribution network (reviewed in Section 50.3)—a circuit structure that does not perform any computational function.

- The reliable operation of the system depends upon the assumptions concerning the value of the propagation delays, which, if not satisfied, can lead to catastrophic timing violations and render the system unusable.

50.3 Synchronous Timing and Clock Distribution Networks

The timing of a synchronous VLSI system is characteristically analyzed at the level of its synchronous building blocks, the local data paths. In the following sections, the simple and effective modeling of synchronous building blocks is described. The effects of clock distribution networks on circuit operation are also presented.

50.3.1 Background

As described in Section 50.2, most high-performance digital integrated circuits implement data-processing algorithms based on the iterative execution of basic operations. Typically, these algorithms are highly *parallelized* and *pipelined* by inserting clocked registers at specific locations throughout the circuit. The synchronization strategy for these clocked registers in the vast majority of VLSI-based digital systems is a fully synchronous approach. It is not uncommon for the computational process in these systems to be spread over hundreds of thousands of functional logic elements and tens of thousands of registers.

For such synchronous digital systems to function properly, the vast number of switching events require a strict temporal ordering. This strict ordering is enforced by a global synchronization signal known as the *clock signal*. For a fully synchronous system to operate correctly, the clock signal must be delivered to every register at a precise *relative* time. The delivery function is accomplished by a circuit and interconnect structure known as a *clock distribution network* [15].

Multiple factors affect the propagation delay of the data signals through the combinational logic gates and the interconnect. Since the clock distribution network is composed of logic gates and interconnection wires, the signals in the clock distribution network are also delayed. Moreover, the dependence of the correct operation of a system on the signal delay in the clock distribution network is far greater than on the delay of the logic gates. Recall that by delivering the clock signal to registers at precise times, the clock distribution network essentially quantizes the time of a synchronous system (into clock periods), thereby permitting the simultaneous execution of operations.

The nature of the on-chip clock signal has become a primary factor in limiting circuit performance, causing the clock distribution network to become a performance bottleneck for high-speed VLSI systems. The primary source of load for the clock distribution network has shifted from the logic gates to the *interconnect*, thereby changing the physical nature of the load from a lumped capacitance (C) to a distributed *resistive-capacitive* (RC) load and eventually a distributed *resistive-capacitive-inductive* (RLC) load [7,16,17]. These interconnect impedances degrade the on-chip signal waveform shapes and increase the path delay. Furthermore, uncertainty is introduced into the signal timing due to statistical variations in the parameters characterizing the circuit elements along the clock and data signal paths, caused by the imperfect control of the manufacturing process and the environment. These changes in circuit behavior have a profound impact on both the choice of synchronous design methodology and on the overall circuit performance. Among the most important consequences are increased power dissipated by the clock distribution network as well as the increasingly challenging timing constraints that must be satisfied to avoid timing violations [3–6,15,18–20].

50.3.2 Definitions and Notation

A synchronous digital system is a network of logic gates and registers whose input and output terminals are interconnected by wires. A sequence of connected logic gates (no registers) is called a signal path. Signal paths bounded by registers are called sequentially adjacent paths.

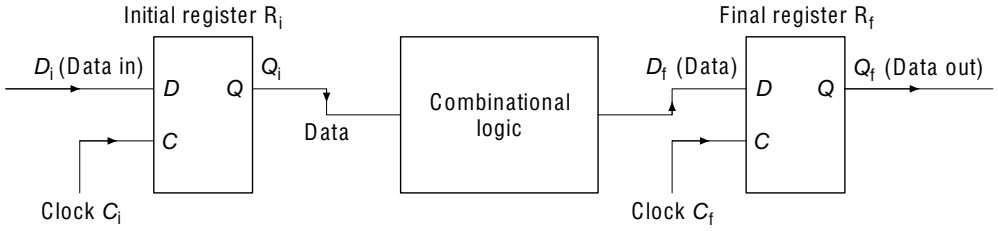


FIGURE 50.3 A local data path.

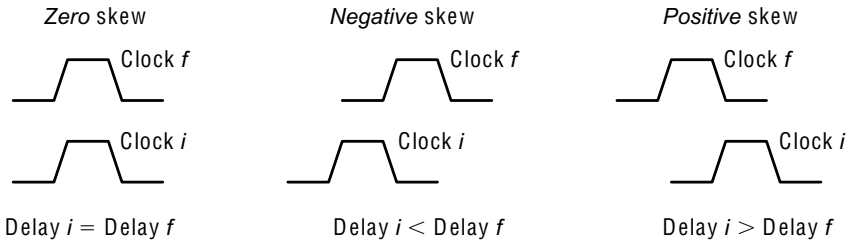


FIGURE 50.4 Lead/lag relationships causing clock skew to be zero, negative, or positive.

Definition 50.1 (Sequentially adjacent pair of registers). For an arbitrary *ordered* pair of registers $\langle \mathsf{R}_i, \mathsf{R}_f \rangle$ in a synchronous circuit, one of the following two situations can be observed. Either there exists at least one signal path that connects some output of R_i to some input of R_f or any input of R_f *cannot* be reached from any output of R_i by propagating through a sequence of logic elements *only.* In the former case—denoted by $\mathsf{R}_1 \rightsquigarrow \mathsf{R}_2$—the pair of registers $\langle \mathsf{R}_i, \mathsf{R}_f \rangle$ is called a *sequentially adjacent pair of registers* and switching events at the output of R_i can possibly affect the input of R_f during the same clock period. A sequentially adjacent pair of registers is also referred to as a *local data path* [15].

A sample local data path with a register (a flip-flop or a latch) is shown in Figure 50.3.[δ] In Figure 50.3, the clock signals C_i and C_f driving the initial register R_i and the final register R_f, respectively, of the local data path are shown.

Definition 50.2 For any ordered pair of registers $\langle \mathsf{R}_i, \mathsf{R}_j \rangle$ in a fully synchronous circuit driven by the clock signals C_i and C_j, respectively, the clock skew $T_{\text{Skew}}(i, j)$ is defined as the difference:

$$T_{\text{Skew}}(i, j) = t^{\text{i}}_{cd} - t^{\text{j}}_{cd} \tag{50.1}$$

where t^{i}_{cd} and t^{j}_{cd} are the clock delays of the clock signals C_i and C_j, respectively.

In Definition 50.2, the clock delays t^{i}_{cd} and t^{j}_{cd} are with respect to some reference point. A commonly used reference point is the source of the clock distribution network. Note that the clock skew $T_{\text{Skew}}(i, j)$ as defined in Definition 50.2 obeys the antisymmetric property,

$$T_{\text{Skew}}(i, j) = -T_{\text{Skew}}(j, i) \tag{50.2}$$

Depending on the values of t^{i}_{cd} and t^{f}_{cd}, the skew can be *zero* ($t^{\text{i}}_{cd} = t^{\text{f}}_{cd}$), *negative* ($t^{\text{i}}_{cd} < t^{\text{f}}_{cd}$), or *positive* ($t^{\text{i}}_{cd} > t^{\text{f}}_{cd}$). This behavior is illustrated in Figure 50.4.

[δ]Examples of local data paths with flip-flops and latches are shown in Figure 50.17 and Figure 50.21, respectively.

Note that the clock skew as defined above is only defined for sequentially-adjacent registers, that is, for local data paths [such as the path shown in Figure 50.2(b)].

Definition 50.3 (Data propagation time). For any arbitrary pair of registers $\langle R_i, R_f \rangle$ in a local data path $R_i \rightsquigarrow R_f$ of a synchronous circuit, the amount of time a data signal is processed in the combinational logic block is defined as the data propagation time $D_P^{i,f}$.

Conventionally, the timing analysis of sequential circuits is performed when the circuit components are modeled with min–max timing models. In the min–max timing model, the delay information of a circuit component is represented with two quantities; the minimum corresponding to the delay of the component under best-case operation conditions and the maximum for the worst-case operation conditions. The subscripts m and M appended to the parameter $D_P^{i,f}$ represent the minimum and maximum data propagation times, $D_{Pm}^{i,f}$ and $D_{PM}^{i,f}$, respectively, constituting the min–max timing model for the local data path $R_i \rightsquigarrow R_f$.

A fully synchronous digital circuit is formally defined as follows:

Definition 50.4 A fully synchronous digital circuit $S = \langle G, R, C \rangle$ is an ordered triple, where

- $G = \{g_1, g_2, \ldots, g_M\}$ is the set of all combinational logic gates,
- $R = \{R_1, R_2, \ldots, R_N\}$ is the set of all registers, and
- $C = \| c_{i \times j} \|_{N \times N}$ is a matrix describing the connectivity of G where for every element $c_{i,j}$ of C

$$c_{i,j} = \begin{cases} 0, & \text{if } R_i \rightsquigarrow R_j \\ 1, & \text{if } R_i \not\rightsquigarrow R_j \end{cases}$$

Note that in a fully synchronous digital system there are no purely combinational signal *cycles*, that is, the input of any logic gate G_k cannot be reached by starting at the same gate and propagating through a sequence of combinational logic gates only [15,21].

50.3.3 Graph Model of a Fully Synchronous Digital Circuit

Certain properties of a synchronous digital circuit may be better understood by analyzing a graph model of a circuit. A synchronous digital circuit can be modeled as a *directed graph* [22,23] G with a *vertex set* $V = \{v_1, \ldots, v_N\}$ and an *edge set* $E = \{e_1, \ldots, e_{N_P}\} \subseteq V \times V$. An example of a circuit graph G is illustrated in Figure 50.5(a). The number of registers in the circuit is $|V| = N$ where the vertex v_k corresponds to the register R_k. The number of local data paths in the circuit is $|E| = N_P = 11$ for the example shown in Figure 50.5. An edge is directed from v_i to v_j iff $R_i \rightsquigarrow R_j$. In the case where multiple paths between a sequentially adjacent pair of registers $R_i \rightsquigarrow R_j$ exist, only *one* edge connects v_i to v_j. The *underlying* graph

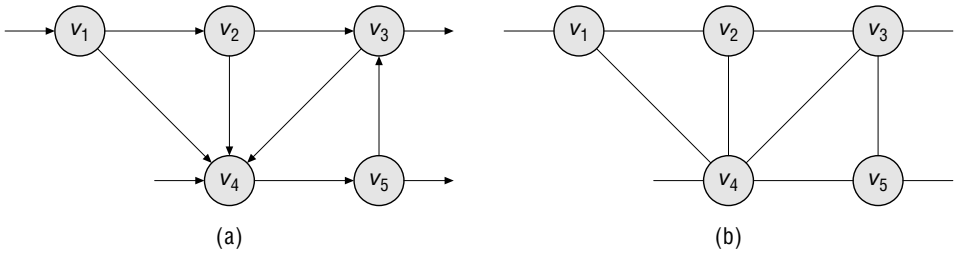


FIGURE 50.5 Graphs G and its underlying graph G_u of a circuit with $N = 5$ registers. (a) The directed graph G. (b) The underlying graph G_u of G.

G_u of the graph G is a *nondirected* graph that has the same vertex set V, where the directions have been removed from the edges. The underlying graph G_u of the graph G depicted in Figure 50.5(a) is shown in Figure 50.5(b). Furthermore, an input or an output of the circuit is indicated in Figure 50.5 by an edge incident to only one vertex.

50.3.4 Clock Skew Scheduling

The majority of the approaches used to design a clock distribution network simplify the performance goals by targeting minimal or zero global clock skew [24–26], which can be achieved by different routing strategies [27–30], buffered clock tree synthesis, symmetric n-ary trees [31] (most notably H-trees), or a distributed series of buffers connected as a mesh [15,32]. A zero clock skew scheme is established by distributing the clock signal to all synchronous components of a circuit with identical clock delays. In other words, the clock skew evaluates to zero on all of the local data paths of a zero clock skew circuit:

$$\forall \left\langle \mathsf{R}_\mathrm{i}, \mathsf{R}_\mathrm{f} \right\rangle : t_{cd}^{\mathrm{i}} = t_{cd}^{\mathrm{f}} \;\Rightarrow\; T_{\mathrm{Skew}}(i, f) = 0 \tag{50.3}$$

For zero clock skew systems, the clock period T_{CP} is limited by the largest maximum data propagation time $D_{PM}^{\mathrm{i,f}}$ on the circuit:

$$\min T_{CP} = \max_{\forall \langle \mathsf{R}_\mathrm{i}, \mathsf{R}_\mathrm{f} \rangle} (D_{PM}^{\mathrm{i,f}}) \tag{50.4}$$

If the circuit operates at any clock period less than the largest maximum data propagation time, a *timing hazard* occurs. For any clock period greater than this value, the circuit is fully functional (no timing hazards occur). Finding a clock period T_{CP} for which a zero clock skew circuit is fully functional (equal to or greater than the largest data propagation time $D_{PM}^{\mathrm{i,f}}$), is always possible, making it convenient to design zero clock skew systems. Consequently, the application of zero clock skew schemes has been central to the design of fully synchronous digital circuits for decades [15,33].

The vector column of clock delays $\mathrm{T_{CD}} = [t_{cd}^1, t_{cd}^2, \ldots]^T$ is called a *clock schedule* [15,34]. A clock schedule that satisfies Eq. (50.3) is called a *trivial* clock schedule. Note that a trivial clock schedule $\mathrm{T_{CD}}$ implies global *zero* clock skew since for any i and f, $t_{cd}^{\mathrm{i}} = t_{cd}^{\mathrm{f}}$, thus, $T_{\mathrm{Skew}}(i, f) = 0$. If $\mathrm{T_{CD}}$ is chosen such that the timing constraints of a circuit are satisfied for every local data path $\mathsf{R}_\mathrm{i} \rightsquigarrow \mathsf{R}_\mathrm{f}$, $\mathrm{T_{CD}}$ is called a *consistent* clock schedule.

The goal of nonzero clock skew scheduling is to compute a consistent clock schedule that is not trivial, while improving the circuit performance. It has been shown in Refs. [15,24–26,35–37] that by adopting a nonzero clock skew synchronization scheme, synchronous circuits can operate at clock periods less than the largest maximum data propagation time of the circuit. In nonzero clock skew systems, the clock signal delays t_{cd} at certain registers are intentionally delayed to provide additional data-processing time on slower local data paths. Mathematically, the nonzero clock skew values (also called *useful skew*) evaluate to $T_{\mathrm{Skew}}(i, f) \neq 0$ for some (or *all*) local data paths $\mathsf{R}_\mathrm{i} \rightsquigarrow \mathsf{R}_\mathrm{f}$ of the circuit.

The process of determining a consistent clock schedule $\mathrm{T_{CD}}$ can be considered as the mathematical problem of optimizing the circuit performance under the timing constraints of a circuit. However, there are important practical issues to consider before a clock schedule can be properly implemented. A clock distribution network must be synthesized such that the clock signal is delivered to each register with the proper delay so as to satisfy the clock skew schedule $\mathrm{T_{CD}}$. Furthermore, this clock distribution network must be constructed so as to minimize the deleterious effects of *interconnect impedances* and *process parameter variations* on the implemented clock schedule. Synthesizing the clock distribution network typically consists of determining a *topology* for the network, together with the circuit design and physical layout of the *buffers* and *interconnect* within the clock distribution network [15].

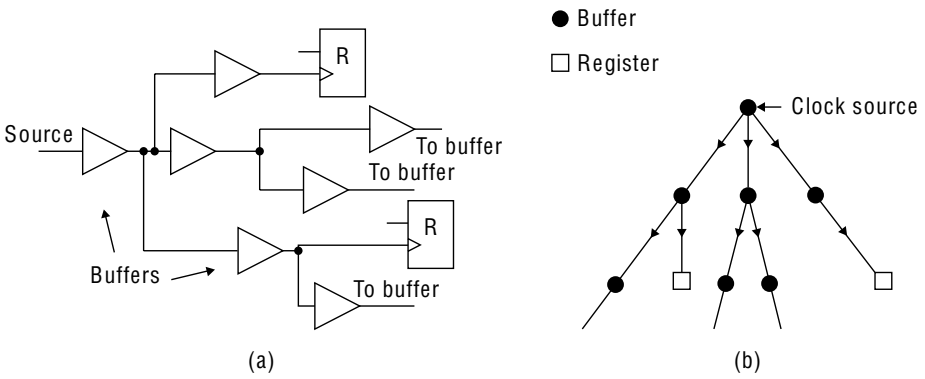


FIGURE 50.6 Tree structure of a clock distribution network. (a) Circuit structure of the clock distribution network. (b) Equivalent graph of a clock tree structure that corresponds to the circuit shown in (a).

50.3.5 Structure of a Clock Distribution Network

The clock distribution network is frequently organized as a rooted tree structure [15,22,24], as illustrated in Figure 50.6, and is often called a *clock tree* [15]. A circuit schematic of a clock distribution network is shown in Figure 50.6(a). An abstract graphical representation of the tree structure depicted in Figure 50.6(a) is shown in Figure 50.6(b). The unique source of the clock signal is at the root of the tree. This signal is distributed from the source to every register in the circuit through a sequence of buffers and interconnect. Typically, a buffer in the network drives a combination of other buffers and registers in the VLSI circuit. An interconnection network of wires connects the output of the driving buffer to the inputs of these driven buffers and registers. An *internal node* of the tree corresponds to a buffer and a *leaf node* of the tree corresponds to a register. There are N leaves[¶] in the clock tree labeled F_1 through F_N where leaf F_j corresponds to register R_j. A clock tree topology that implements a given clock schedule T_{CD} must enforce a clock skew $T_{Skew}(i, f)$ for each local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ of the circuit to ensure that the timing constraints of the circuit are satisfied. This topology, however, can be affected by three important issues relating to the operation of a fully synchronous digital system.

50.3.5.1 Linear Dependency of the Clock Skews

An important corollary related to the *conservation property* [15] of clock skew is that there exists a *linear dependency* among the clock skews of a global data path that form a cycle in the underlying graph of the circuit. Specifically, if $v_0, e_1, v_1 (\neq v_0), \ldots, v_{k-1}, e_k, v_k \equiv v_0$ is a cycle in the underlying graph of the circuit,

$$0 = [t_{cd}^0 - t_{cd}^1] + [t_{cd}^1 - t_{cd}^2] + \cdots = \sum_{i=0}^{k-1} T_{Skew}(i, i+1) \tag{50.5}$$

The property described by Eq. (50.5) is illustrated in Figure 50.5 for the undirected cycle v_1, v_4, v_3, v_2, v_1. Note that

$$\begin{aligned} 0 &= (t_{cd}^1 - t_{cd}^4) + (t_{cd}^4 - t_{cd}^3) + (t_{cd}^3 - t_{cd}^2) + (t_{cd}^2 - t_{cd}^1) \\ &= T_{Skew}(1,4) + T_{Skew}(4,3) + T_{Skew}(3,2) + T_{Skew}(2,1) \end{aligned} \tag{50.6}$$

[¶]The number of registers N in the circuit.

The importance of this property is that Eq. (50.5) describes the inherent correlation among certain clock skews within a circuit. These correlated clock skews therefore *cannot* be independently optimized. Returning to Figure 50.5, note that it is not necessary that a directed cycle exists in the directed graph G of a circuit for Eq. (50.5) to hold. For example, v_2,v_3,v_4 is not a cycle in the directed circuit graph G in Figure 50.5(a) but v_2,v_3,v_4 is a cycle in the undirected circuit graph G_u in Figure 50.5(b). In addition, $T_{\text{Skew}}(2,3) + T_{\text{Skew}}(3,4) + T_{\text{Skew}}(4,2) = 0$, that is, the skews $T_{\text{Skew}}(2,3)$, $T_{\text{Skew}}(3,4)$, and $T_{\text{Skew}}(4,2)$ are linearly dependent. A maximum of $(|V| - 1) = (N-1)$ clock skews can be chosen independently of each other in a circuit, which is easily proven by considering a spanning tree of the underlying circuit graph G_u [22,23]. Any spanning tree of G_u will contain $(N - 1)$ edges—each edge corresponding to a local data path—and the addition of any other edge of G_u will form a cycle such that Eq. (50.5) holds for this cycle. Note, for example, that for the circuit modeled by the graph shown in Figure 50.5, *four* independent clock skews can be chosen such that the remaining three clock skews can be expressed in terms of the independent clock skews.

The interdependency of the clock skew values makes the analysis of clock skew scheduling methods a difficult problem. Owing to this interdependency characteristic, a clock skew value cannot be determined independent of the remaining clock skews, thus a typical clock-skew scheduling method must simultaneously encompass the analysis of all local data paths. Such simultaneous analysis of all local data paths in a given synchronous circuit is typically structured by including the timing constraints of every local data path in a single optimization problem.

50.3.5.2 Differential Character of the Clock Tree

In a given circuit, the clock signal delay t_{cd}^{j} from the clock source to the register R_j is equal to the sum of the propagation delays of the buffers on the unique path that exists between the root of the clock tree and the leaf F_j corresponding to the jth register. Furthermore, if $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ is a sequentially adjacent pair of registers, there is a portion of the two paths—denoted P_{if}^{*}—between the root of the clock tree and R_i and R_f, respectively, that is common to both paths. This concept is illustrated in Figure 50.7. A portion of a clock tree is shown in Figure 50.7 where each of the vertices 1 through 9 corresponds to a buffer in the clock tree. The vertices 4, 5, and 9 are leaves of the tree and correspond to the registers R_4, R_5, and

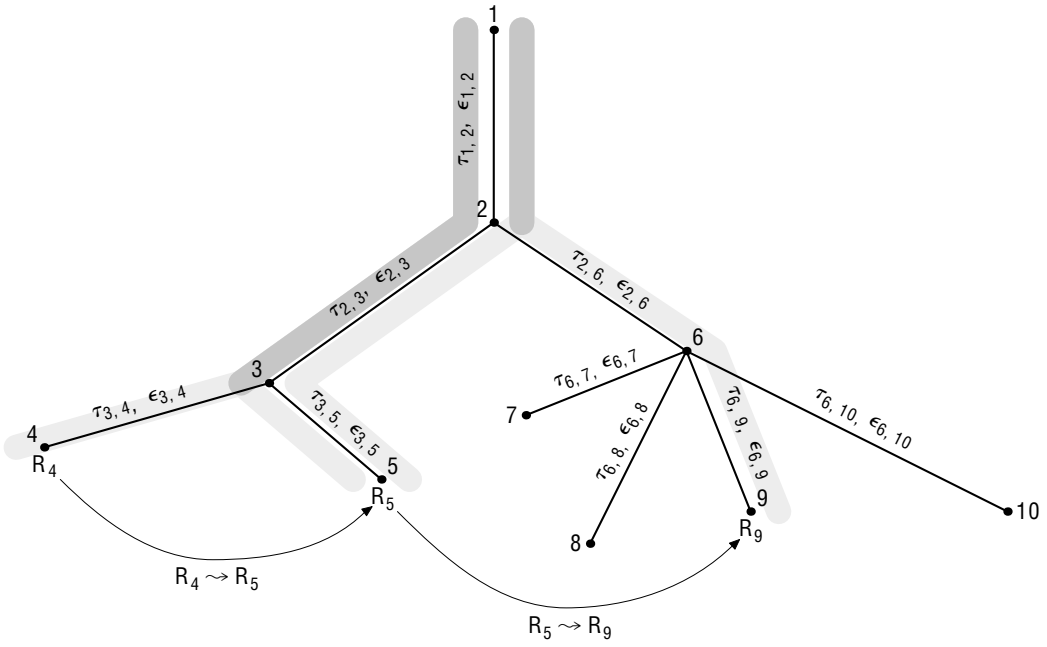


FIGURE 50.7 Illustration of the differential nature of the clock tree.

TABLE 50.1 Target and Actual Values of the Clock Skews for the Local Data Paths $R_4 \rightsquigarrow R_5$ and $R_5 \rightsquigarrow R_9$ Shown in Figure 50.7

	Target Skew	Actual Skew Bounds
$T_{\text{Skew}}(4,5)$	$\tau_{3,4} - \tau_{3,5}$	$\tau_{3,4} - \tau_{3,4} \pm (\epsilon_{3,4} + \epsilon_{3,4})$
$T_{\text{Skew}}(5,9)$	$\tau_{2,3} + \tau_{3,5} - \tau_{2,6} - \tau_{6,9}$	$\tau_{2,3} + \tau_{3,5} - \tau_{2,6} - \tau_{6,9} \pm (\epsilon_{2,3} + \epsilon_{3,5} + \epsilon_{2,6} + \epsilon_{6,9})$

R_9, respectively.[‖] The local data paths $R_4 \rightsquigarrow R_5$ and $R_5 \rightsquigarrow R_9$ are indicated in Figure 50.7 with arrows while the paths of the clock signals to each of the registers R_4, R_5, and R_9 are shown in Figure 50.7 as lightly shaded. The portion of the clock signal paths common to both registers of a local data path is shaded darker in Figure 50.7—note the segments $1 \rightarrow 2 \rightarrow 3$ for $R_4 \rightsquigarrow R_5$ and $1 \rightarrow 2$ for $R_5 \rightsquigarrow R_9$.

Similarly, there is a portion of the clock signal path to any of the registers R_i and R_f in a sequentially adjacent pair of registers $R_i \rightsquigarrow R_f$, denoted by P^i_{if} and P^f_{if}, respectively, that is unique to this register. Returning to Figure 50.7, the segments $3 \rightarrow 4$ and $3 \rightarrow 5$ are unique to the clock signal paths to the registers R_4 and R_5 while the segments $2 \rightarrow 3 \rightarrow 5$ and $2 \rightarrow 6 \rightarrow 9$ are unique to the clock signal paths to the registers R_5 and R_9, respectively.

Note that the clock skew $T_{\text{Skew}}(i,f)$ between the sequentially adjacent pair of registers $R_i \rightsquigarrow R_f$ is equal to the difference between the accumulated buffer propagation delays between P^i_{if} and P^f_{if}, that is, $T_{\text{Skew}}(i,f) = \text{Delay}(P^i_{if}) - \text{Delay}(P^f_{if})$. Therefore, any variation of circuit parameters over P^*_{if} will not affect the value of the clock skew $T_{\text{Skew}}(i,f)$. For the example shown in Figure 50.7, $T_{\text{Skew}}(4,5) = \text{Delay}(P^4_{4,5}) - \text{Delay}(P^5_{4,5})$ and $T_{\text{Skew}}(5,9) = \text{Delay}(P^5_{5,9}) - \text{Delay}(P^9_{5,9})$.

This differential feature of the clock tree suggests an approach for minimizing the effects of process parameter variations on the correct operation of the circuit. To illustrate this approach, each branch $p \rightarrow q$ of the clock tree shown in Figure 50.7 is labeled with two numbers—$\tau_{p,q} > 0$ is the intended delay of the branch and $\epsilon_{p,q} \geq 0$ is the maximum error (deviation) of this delay.[**] In other words, the actual delay of the branch $p \rightarrow q$ is in the interval $[\tau_{p,q} - \epsilon_{p,q}, \tau_{p,q} + \epsilon_{p,q}]$. With this notation, the *target* clock skew values for the local data paths $R_4 \rightsquigarrow R_5$ and $R_5 \rightsquigarrow R_9$ are shown in the middle column in Table 50.1. The bounds of the actual clock skew values for the local data paths $R_4 \rightsquigarrow R_5$ and $R_5 \rightsquigarrow R_9$ (considering the ϵ variations) are shown in the rightmost column in Table 50.1.

As the results listed in Table 50.1 demonstrate, it is advantageous to maximize P^*_{if} for any local data path $R_i \rightsquigarrow R_f$ such that the parameter variations on P^*_{if} do not affect $T_{\text{Skew}}(i,f)$.

50.4 Timing Properties of Synchronous Storage Elements

The general structure and principles of operation of a fully synchronous digital VLSI system are described in Section 50.2. In this section, the timing constraints due to the combinational logic and the storage elements within a synchronous system are reviewed. The clock distribution network provides the time reference for the storage elements—or registers—thereby enforcing the required logical order of operations. This time reference consists of one or more clock signals that are delivered to each and every register within the integrated circuit. These *clock* signals control the order of computational events by controlling the exact times the register data inputs are sampled.

The data signals are inevitably delayed as these signals propagate through the logic gates and along interconnections within the local data paths. These propagation delays can be evaluated within a certain accuracy (deterministically or statistically) and used to derive timing relationships among signals in a

[‖]Note that *not* all of the vertices correspond to registers.

[**]The deviation ϵ is due to parameter variations during ciruit manufacturing as well as to environmental conditions during operation of the circuit.

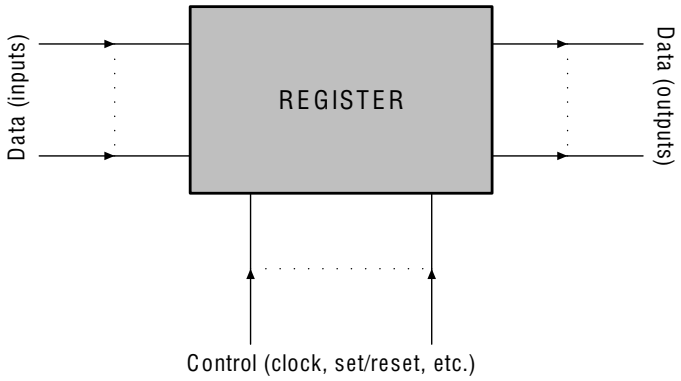


FIGURE 50.8 A general view of a register.

circuit. In this section, the properties of commonly used types of registers and their local timing relationships for different types of local data paths are described. After discussing registers in general in Section 50.4.1, the properties of level-sensitive registers (latches) and the significant timing parameters of these registers are reviewed in Sections 50.4.2 and 50.4.3, respectively. Edge-triggered registers (flip-flops) and related timing parameters are analyzed in Sections 50.4.4 and 50.4.5, respectively. Properties and definitions related to the clock distribution network are reviewed in Section 50.4.6. Finally, the mathematical foundation for analyzing timing violations in flip-flops and latches are discussed in Sections 50.4.7 and 50.4.8, respectively.

50.4.1 Storage Elements

The storage elements (registers) encountered throughout VLSI systems vary widely in function and temporal relationships. Independent of these differences, however, all storage elements share a common feature—the existence of two groups of signals with largely different purpose. A generalized view of a register is depicted in Figure 50.8. The I/O signals of a register can be divided into two groups as shown in Figure 50.8. One group of signals—called the *data* signals—consists of input and output signals of the storage element. These input and output signals are connected to the *data* signal terminals of other storage elements as well as to the terminals of ordinary logic gates. Another group of signals—identified by the name *control* signals—are those signals that control the storage of the data signals in the registers but do not participate in the logical computation process.

Certain control signals enable the storage of a data signal in a register independently of the values of any data signals. These control signals are typically used to initialize the data in a register to a specific well-known value. Other control signals—such as a *clock* signal—control the process of storing a data signal within a register. In a synchronous circuit, each register has at least one clock (or control) signal input.

The two major groups of storage elements (registers) are considered in the following sections based on the type of relationship that exists between the data and clock signals of these elements. In *latches*, it is the specific value or level of a control signal[*†] that determines the data storage process. Therefore, latches are also called *level-sensitive registers*. In contrast to latches, a data signal is stored in *flip-flops*, controlled by an *edge* of a control signal. For that reason, flip-flops are also called *edge-triggered registers*. The timing properties of latches and flip-flops are described in the following sections.

50.4.2 Latches

A *latch* is a register whose behavior depends upon the value or level of the clock signal [9,38–44]. A latch is therefore often referred to as a *transparent* latch, a *level-sensitive* register, or a *polarity hold* latch. A simple

[*†] This signal is most frequently the clock signal.

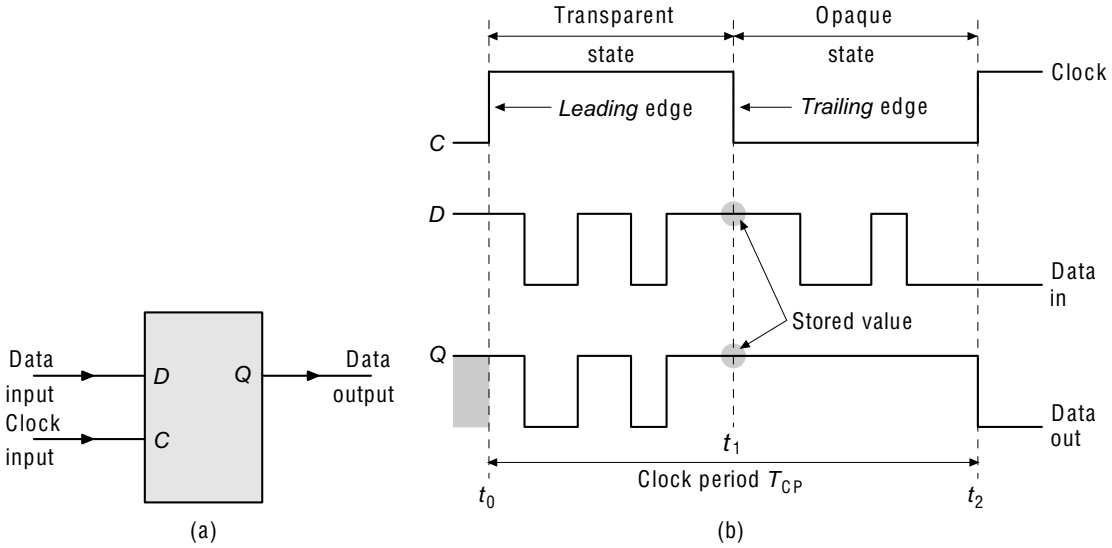


FIGURE 50.9 Schematic representation and principle of operation of a level-sensitive register (latch). (a) A level-sensitive register or latch. (b) Idealized operation of the latch shown in (a).

TABLE 50.2 Operation of the Positive-Polarity *D* Latch

Clock	Output	State
High	Passes input	Transparent
Low	Maintains output	Opaque

type of latch with a clock signal C and an input signal D is depicted in Figure 50.9(a)—the output of the latch is typically labeled Q. This type of latch is also known as a D latch. The operation of a D latch is illustrated in Figure 50.9(b).

The register illustrated in Figure 50.9 is a *positive-polarity*[*‡] latch since the register is transparent during that portion of the clock period for which C is high. The operation of this positive latch is summarized in Table 50.2.

As described in Table 50.2 and illustrated in Figure 50.9(b), the output signal of the latch follows the data input signal while the clock signal remains high, i.e., $C = 1 \Rightarrow Q = D$. Therefore, the latch is said to be in a *transparent* state during the interval $t_0 < t < t_1$ as shown in Figure 50.9(b). When the clock signal C changes from 1 to 0, the current value of D is stored in the register and the output Q remains fixed to that value regardless of whether the data input D changes. The latch does *not* pass the input data signal to the output, but rather holds onto the last value of the data signal when the clock signal made the high-to-low transition. By analogy with the term *transparent* introduced above, this state of the latch is called *opaque* and corresponds to the interval $t_1 < t < t_2$ as shown in Figure 50.9(b) where the input data signal is isolated from the output port. As shown in Figure 50.9(b), the clock period is $T_{CP} = t_2 - t_0$.

The edge of the clock signal that causes the latch to switch to the transparent state is identified as the *leading* edge of the clock pulse. In the case of the positive latch shown in Figure 50.9(a), the leading edge of the clock signal occurs at time t_0. The opposite direction edge of the clock signal is identified as the *trailing* edge—the falling edge at time t_1 shown in Figure 50.9(b). Note that for a negative latch, the leading edge is a high-to-low transition and the trailing edge is a low-to-high transition.

[*‡] Or simply a *positive* latch.

50.4.3 Parameters of Latches

Registers such as the *D* latch illustrated in Figure 50.9 and the flip-flops described in Sections 50.4.4 and 50.4.5 are built of discrete transistors. The exact relationships among the signals on the terminals of a register can be presented and evaluated in analytic form [45–47]. In this section, however, registers are considered at a higher level of abstraction to hide the details of the specific electrical implementation. The latch parameters are briefly introduced next.

Note that the remaining portion of this section uses an extensive notation for various parameters of signals and storage elements. A glossary of terms used throughout this section is listed in the appendix.

50.4.3.1 Minimum Width of the Clock Pulse

The *minimum width of the clock pulse* C^L_{Wm} is the *minimum* permissible width of this portion of the clock signal during which the latch is transparent. In other words, C^L_{Wm} is the length of the time interval between the leading and the trailing edge of the clock signal such that the latch will operate properly. The minimum width of the clock pulse is determined by multiple factors, including the technological limitations of the manufacturing process and the clock signal generation circuit. Further increasing the value of C^L_{Wm} will *not* affect the values of D^L_{DQ}, δ^L_S, and δ^L_H (defined in Sections 50.4.3.3, 50.4.3.4, and 50.4.3.5, respectively). The minimum width of the clock pulse, $C^L_{Wm} = t_6 - t_1$, is illustrated in Figure 50.10. The clock period is $T_{CP} = t_8 - t_1$.

50.4.3.2 Latch Clock-to-Output Delay

The *clock-to-output delay* D^L_{CQ} (typically called the clock-to-*Q* delay) is the propagation delay of the latch from the *clock* signal terminal to the output terminal. The value of $D^L_{CQ} = t_2 - t_1$ is depicted in Figure 50.10 and is defined assuming that the *data* input signal has settled sufficiently early to a stable value. Setting the data input signal earlier with respect to the leading clock edge will not affect the value of D^L_{CQ}.

50.4.3.3 Latch Data-to-Output Delay

The *data-to-output delay* D^L_{DQ} (typically called the data-to-*Q* delay) is the propagation delay of the latch from the *data* signal terminal to the output terminal. The value of D^L_{DQ} is determined assuming that the clock signal has set the latch to the transparent state sufficiently early. Making the leading edge of the

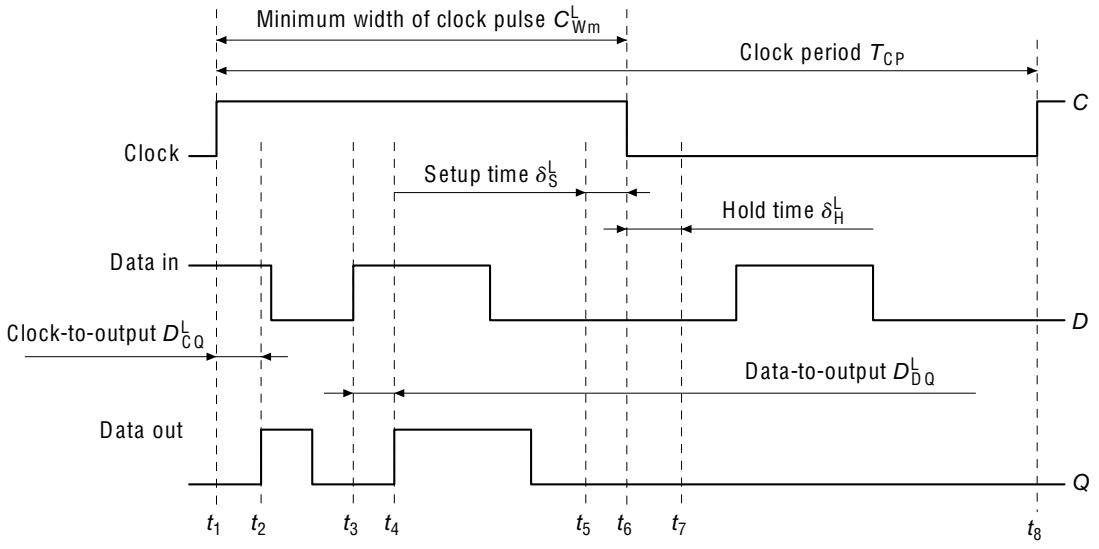


FIGURE 50.10 Parameters of a level-sensitive register.

clock signal occur earlier will not change the value of D^L_{DQ}. The data-to-output delay $D^L_{DQ} = t_4 - t_3$ is illustrated in Figure 50.10.

50.4.3.4 Latch Setup Time

The *latch setup time* $\delta^L_S = t_6 - t_5$, shown in Figure 50.10, is the *minimum* time between a change in the data signal and the trailing edge of the clock signal such that the new value of D would propagate to the output Q of the latch and be stored within the latch during the opaque state.

50.4.3.5 Latch Hold Time

The *latch hold time* δ^L_H is the minimum time after the trailing clock edge that the data signal must remain constant so that this value of D is successfully stored in the latch during the opaque state. This definition of δ^L_H assumes that the last change of the value of D has occurred no later than δ^L_S before the trailing edge of the clock signal. The term $\delta^L_H = t_7 - t_6$ is shown in Figure 50.10.

Sections 50.4.3.1 through 50.4.3.5 are used to refer to any latch in general or, to a specific instance of a latch when this instance can be unambiguously identified. To explicitly refer to a specific instance R_i of a latch, the parameters are additionally shown with a superscript. For example, D^{Li}_{CQ} refers to the clock-to-output delay of latch R_i. Also, adding m and M to the subscript of D^L_{CQ} and D^L_{DQ} may be used to refer to the *minimum* and *maximum* values of D^L_{CQ} and D^L_{DQ}, respectively.

50.4.4 Flip-Flops

An *edge-triggered register* or *flip-flop* is a type of register which, unlike the latches described in Sections 50.4.2 and 50.4.3, is never transparent with respect to the input data signal [9,38–44]. The output of a flip-flop normally does not follow the input data signal at any time during the register operation but rather holds onto a previously stored data value until a new data signal is stored in the flip-flop. A simple type of flip-flop with a clock signal C and an input signal D is shown in Figure 50.11(a)—similar to latches, the output of a flip-flop is usually labeled Q. This specific type of register, shown in Figure 50.11(a), is called a D flip-flop. The operation of a D flip-flop is illustrated in Figure 50.11(b).

In typical flip-flops, data is stored either on the rising edge (low-to-high transition) or on the falling edge (high-to-low transition) of the clock signal. The flip-flops are known as *positive-edge-triggered* and *negative-edge-triggered* flip-flops, respectively. The term *latching*, *storing*, or *positive* edge is used to identify the edge of the clock signal on which storage in the flip-flop occurs. For clarity, the latching edge of the

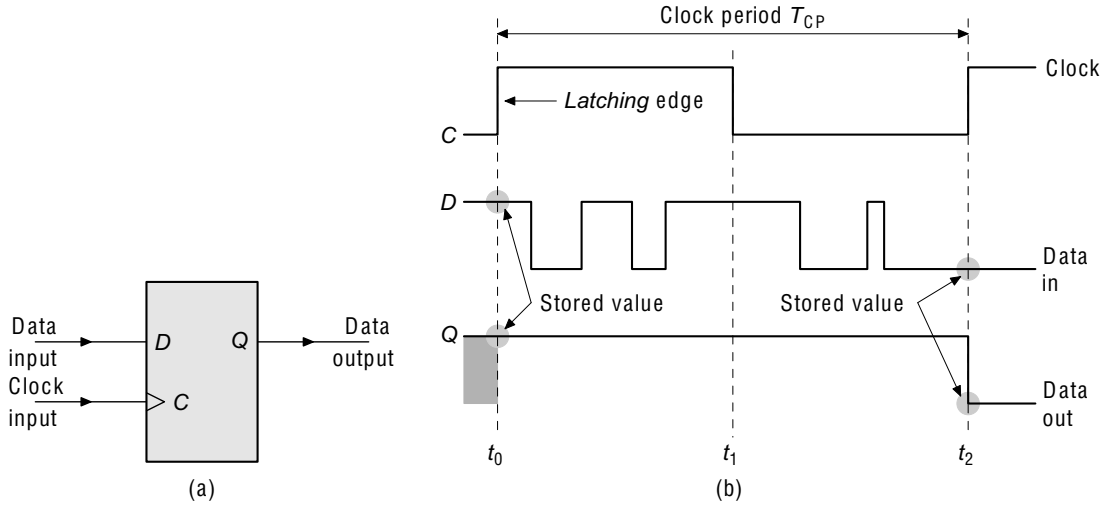


FIGURE 50.11 Schematic representation and principle of operation of an edge-triggered register (flip-flop). (a) An edge-triggered register or a flip-flip. (b) Idealized operation of the flip-flop shown in (a).

clock signal for flip-flops will also be called the leading edge (compare to the discussion of latches in Sections 50.4.2 and 50.4.3). Also note that certain flip-flops—known as *double-edge-triggered* (DET) flip-flops [48–52]—can store data at either edge of the clock signal. The complexity of these flip-flops, however, is significantly higher and the use of DET flip-flops is not very common.

As shown in the timing diagram in Figure 50.11(b), the output of the flip-flop remains unchanged most of the time regardless of the transitions in the data signal. Only values of the data signal in the vicinity of the storing edge of the clock signal can affect the output of the flip-flop. Therefore, changes in the output will only be observed when the currently stored data has a logic value x and the storing edge of the clock signal occurs while the input data signal has a logic value of $\overline{x}$.

50.4.5 Parameters of Flip-Flops

The significant timing parameters of an edge-triggered register are similar to those of latches (recall Section 50.4.3) and are presented next. These parameters are illustrated in Figure 50.12.

50.4.5.1 Minimum Width of the Clock Pulse

The minimum width of the clock pulse C^F_{Wm} is the *minimum* permissible width of the time interval between the latching edge and *non*latching edge of the clock signal. The minimum width of the clock pulse $C^F_{Wm} = t_6 - t_3$ is shown in Figure 50.12 and is the minimum interval between the latching and nonlatching edges of the clock pulse such that the flip-flop will operate correctly. Further increasing C^F_{Wm} will *not* affect the values of the setup time δ^F_S and hold time δ^F_H (defined in Sections 50.4.5.3 and 50.4.5.4, respectively). The clock period $T_{CP} = t_6 - t_1$ is also shown in Figure 50.12.

50.4.5.2 Flip-Flop Clock-to-Output Delay

As shown in Figure 50.12, the *clock-to-output delay* D^F_{CQ} of the flip-flop is $D^F_{CQ} = t_5 - t_3$. This propagation delay parameter—typically called the clock-to-Q delay—is the propagation delay from the clock signal terminal to the output terminal. The value of D^F_{CQ} is defined assuming that the data input signal has settled to a stable value sufficiently early. Setting the data input any earlier with respect to the latching clock edge will not affect the value of D^F_{CQ}.

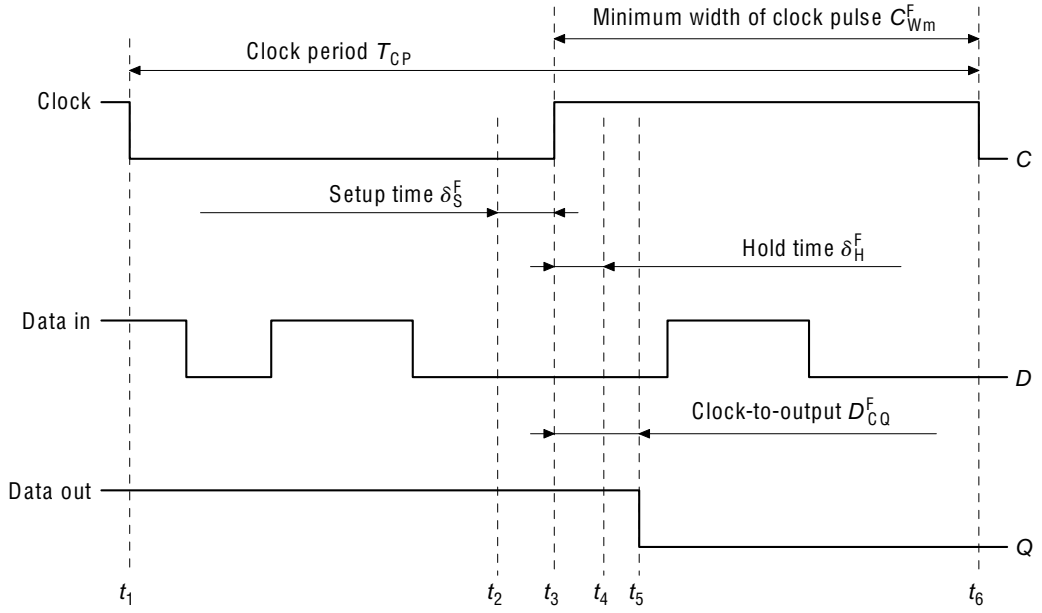


FIGURE 50.12 Parameters of an edge-triggered register.

50.4.5.3 Flip-Flop Setup Time

The flip-flop *setup time* δ_S^F is shown in Figure 50.12 as $\delta_S^F = t_3 - t_2$. The parameter δ_S^F is defined as the *minimum* time between a change in the data signal and the latching edge of the clock signal such that the new value of D propagates to the output Q of the flip-flop and is successfully latched within the flip-flop.

50.4.5.4 Flip-Flop Hold Time

The flip-flop *hold time* δ_H^F is the minimum time after the arrival of the latching clock edge in which the data signal must remain constant to successfully store the D signal within the flip-flop. The hold time $\delta_H^F = t_4 - t_3$ is illustrated in Figure 50.12. This definition of the hold time assumes that the last change of D has occurred no later than δ_S^F before the arrival of the latching edge of the clock signal.

Note that similar to latches, the parameters of these edge-triggered registers refer to any flip-flop in general, or to a specific instance of a flip-flop when this instance is uniquely identified. To explicitly refer to a specific instance i of a flip-flop, the flip-flop parameters are additionally shown with a superscript. For example, δ_S^{Fi} refers to the setup time parameter flip-flop i. Also, adding m and M to the subscript of D_{CQ}^F may be used to refer to the *minimum* and *maximum* values of D_{CQ}^F, respectively.

50.4.6 The Clock Signal

The clock signal is typically delivered to each storage element within a circuit. This signal is crucial to the correct operation of a fully synchronous digital system. The storage elements serve to establish the relative sequence of events within a system such that those operations that cannot be executed concurrently operate sequentially on the proper data signals.

A typical clock signal $c(t)$ in a synchronous digital system is shown in Figure 50.13. The *clock period* T_{CP} of $c(t)$ is indicated in Figure 50.13. To provide the highest possible clock frequency, the objective is for T_{CP} to have the smallest value such that

$$\forall n: \quad c(t) = c(t + nT_{CP}) \tag{50.7}$$

where n is an integer. The width of the clock pulse C_W, shown in Figure 50.13, is explained in Sections 50.4.3.1 and 50.4.5.1.

Typically, the period of the clock signal T_{CP} is a constant, that is, $\partial T_{CP}/\partial t = 0$. If the clock signal $c(t)$ has a delay τ from some reference point, then the leading edges of $c(t)$ occur at times

$$\tau + mT_{CP} \quad \text{for} \quad m \in \{\ldots, -2, -1, 0, 1, 2, \ldots\} \tag{50.8}$$

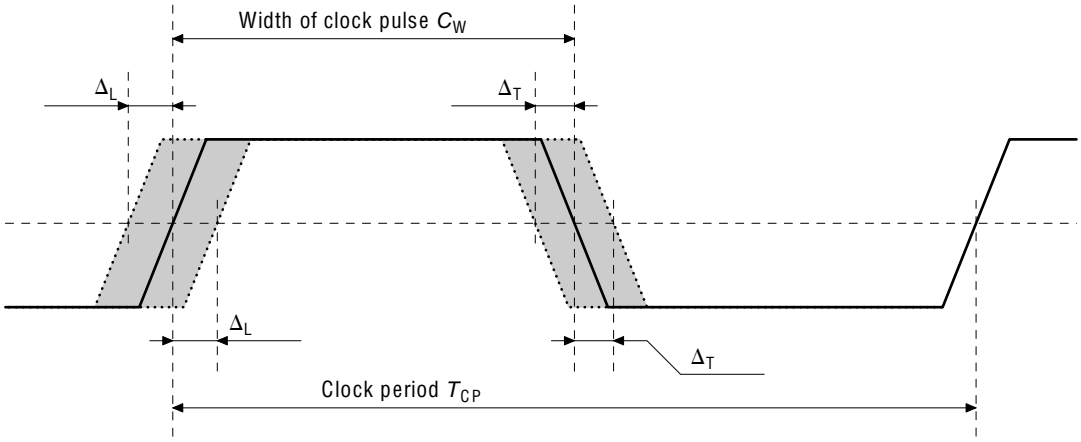


FIGURE 50.13 A typical clock signal.

and the trailing edges of $c(t)$ occur at times

$$\tau + C_W + mT_{CP} \quad \text{for} \quad m \in \{\ldots, -2, -1, 0, 1, 2, \ldots\} \tag{50.9}$$

In practice, however, it is possible for the edges of a clock signal to fluctuate in time, that is, *not* to occur precisely at the times described by Eq. (50.8) and Eq. (50.9) for the leading and trailing edges, respectively. This phenomenon is known as *clock jitter* and may be due to various causes such as variations in the manufacturing process, ambient temperature, power supply noise, and oscillator characteristics.

To account for this clock jitter, the following parameters are introduced:

- The maximum deviation Δ_L of the leading edge of the clock signal such that the leading edge occurs anywhere in an interval $(\tau + kT_{CP} - \Delta_L, \tau + kT_{CP} + \Delta_L)$
- The maximum deviation Δ_T of the trailing edge of the clock signal such that the leading edge occurs anywhere in the interval $(\tau + C_W + kT_{CP} - \Delta_T, \tau + C_W + kT_{CP} + \Delta_T)$

50.4.6.1 Synchronization Schemes

Traditionally, a single-phase clock signal such as the waveform shown in Figure 50.14 is used for synchronization. This relatively easy to implement and analyze single-phase clocking scheme (built with a single-phase clock signal) has several shortcomings in satisfying the timing requirements of circuits manufactured in nanoscale technologies. Under a single-phase clocking scheme, for instance, it becomes infeasible for signals to propagate across the entire area of an integrated circuit within a single clock cycle. To satisfy the increasingly complex timing requirements of integrated circuits, advanced synchronization methodologies such as *multiclock domains* and *multiphase clock signals* are used. These two concepts are briefly reviewed and the operational characteristics are described in this section.

Multiclock domains consist of two or more nonidentical clock signals delivered within a circuit for synchronization. It is possible to use well-tuned clocking schemes within each clock domain, improving the overall operational characteristics of the circuit. The clock signals are typically generated by separate oscillators, thus the frequency and phase information of the clock signals can be independent. The availability of multiple clock signals enables relatively independent synchronization of different domains within a circuit. Communication between (and among) multiple clock domains (the timing of local data paths between multiple clock domains) are managed by simultaneously considering the properties of the multiple clock domains simultaneously.

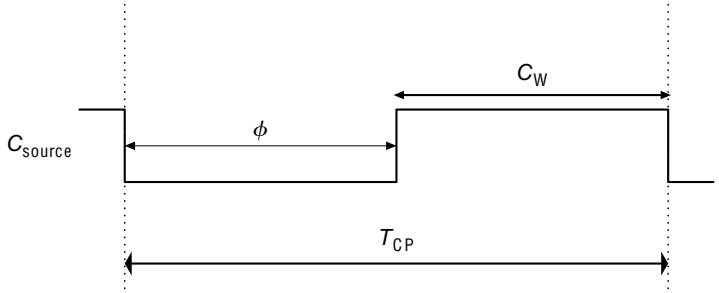


FIGURE 50.14 A single-phase synchronization clock signal.

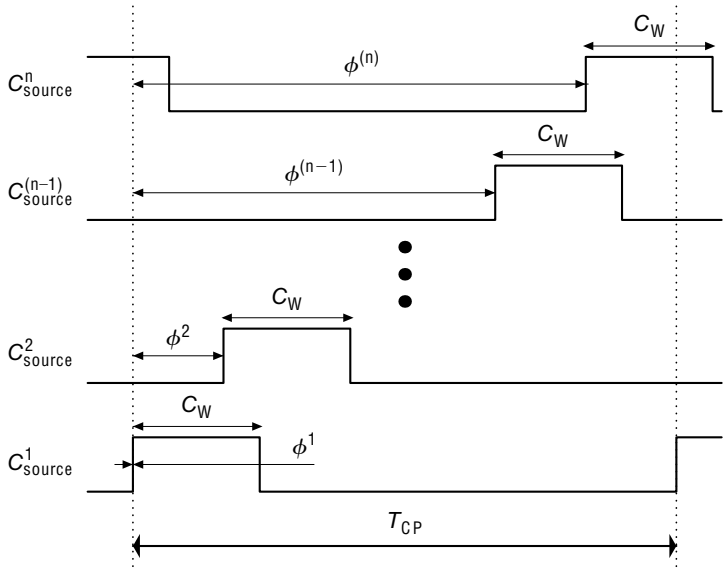


FIGURE 50.15 A generic multiphase synchronization clock.

Multiphase clock signals are generated and used to provide alternate phases of the clock signal for synchronization. Multiphase synchronization clock signals are not as customizable as the clock signals in a multiclock domain application. Multiphase synchronization, however, incurs a smaller design and architecture overhead. Clock waveforms for a multiphase synchronization scheme are shown in Figure 50.15. In Figure 50.15, the set of clock signals $C^{\text{global}} = \{C^1, \ldots, C^m\}$ constitutes the n-phase clocking scheme. The subscripts denote the location of the clock signals in the circuit. For instance, C^1_{source} denotes the clock signal at the clock source of the clock phase C^1. When this clock signal is delivered to an arbitrary register R_k, the (delayed) signal is represented by C^1_k. The start time ϕ^{p_i} of the clock signal phase C^{p_i} is defined with respect to a common reference clock cycle. The *phase shift operator* $\phi^{p_i p_f}$ [53] is used to transform variables among different clock phases. The phase shift operator $\phi^{p_i p_f}$ is defined as the algebraic difference $\phi^{p_i p_f} = \phi^{p_i} - \phi^{p_f} + kT$, where k is the number of clock cycles occurring between phases C^{p_i} and C^{p_f}. Note that for a single-phase clocking scheme, the phase shift operator evaluates to $\phi^{if} = T$.

Dual-phase, nonoverlapping clock signals are easier to generate in practice as compared to generating overlapping phases or arbitrary n-phase clock signals. In the implementation of a dual-phase synchronization scheme, a single-phase clock is typically distributed throughout the circuit and inverted locally to generate the nonoverlapping second phase. Consequently, such dual-phase schemes are more popular as compared to more complex multiphase synchronization schemes. The dual-phase synchronization scheme is particularly popular in level-sensitive circuits. Two-phase, level-sensitive circuits operate similarly to single-phase edge-triggered circuits [54,55], making it possible to implement a logic network with either flip-flop or latch storage elements without major topological changes.

50.4.6.2 Clock Skew in Multiphase Schemes

The definition of clock skew can be extended to those circuits synchronized with a multiphase synchronization scheme (see Section 50.4.6.1). For a multiphase synchronized circuit, `Clock i` and `Clock f` (shown in Figure 50.4) are often clock signals with two different phases. Thus, the delays depicted in Figure 50.4 must consider the *phase* difference $\phi^{p_i p_f}$ between the clock signals, and any existing clock skew. Clock skew in a system synchronized with a multiphase synchronization scheme is illustrated in Figure 50.16. In Figure 50.16, clock skew is shown for the generic multiphase synchronization scheme described in Figure 50.15. The

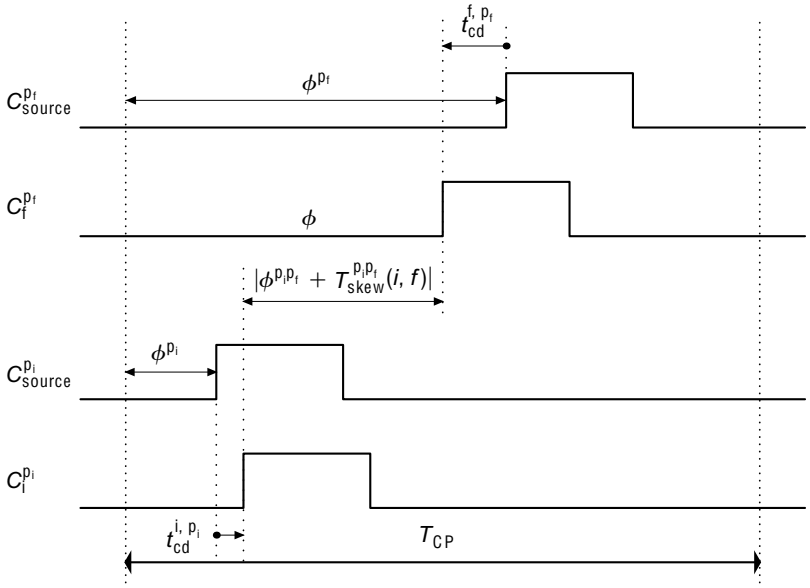


FIGURE 50.16 Multiphase clock skew.

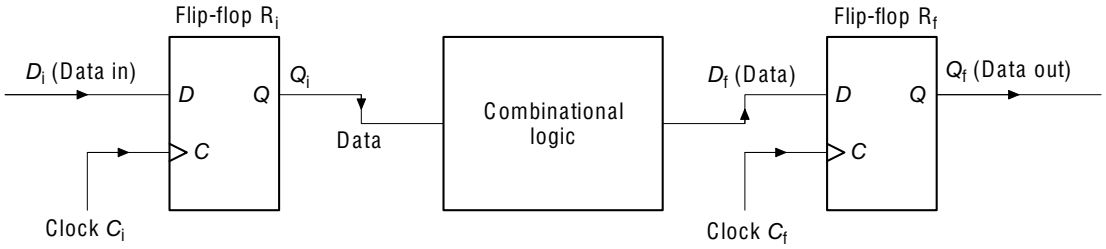


FIGURE 50.17 A single-phase local data path with flip-flops.

parameters t_{cd}^{i,p_i} and t_{cd}^{f,p_f} are the delay of the clock signals $C_i^{p_i}$ and $C_f^{p_f}$ from the clock source to the registers R_i and R_f, respectively. Using this notation, multiphase clock skew is represented as

$$T_{skew}^{p_i p_f}(i, f) = t_{cd}^{i,p_i} - t_{cd}^{f,p_f}$$

In Sections 50.4.7 and 50.4.8, the timing behavior of edge-triggered and level-sensitive circuits, respectively, are analyzed considering a single synchronization clock. The analysis can be systematically extended to circuits synchronized by a multiphase clock, using the multiphase clock skew definition presented above.

50.4.7 Analysis of a Single-Phase Local Data Path with Flip-Flops

A local data path composed of two flip-flops and combinational logic between the flip-flops is shown in Figure 50.17.

An analysis of the timing properties of the local data path shown in Figure 50.17 is presented in the following sections. First, the timing relationships to prevent the late arrival of data signals to R_f are examined in Section 50.4.7.1. The timing relationships to prevent the early arrival of signals to the register R_f are described in Section 50.4.7.2. The analyses presented in Sections 50.4.7.1 and 50.4.7.2 borrow some of the notation from Refs. [13,14]. Similar analyses of synchronous circuits from a timing perspective can be found in Refs. [53,56–59].

50.4.7.1 Preventing the Late Arrival of the Data Signal in a Local Data Path with Flip-Flops

The operation of the local data path $R_i \rightsquigarrow R_f$ as shown in Figure 50.17 requires that any data signal that is stored in R_f arrives at the data input D_f of R_f no later than δ_S^{Ff} before the arrival of the latching edge of the clock signal C_f. It is possible for the opposite event to occur, that is, for the data signal D_f *not* to arrive at the register R_f sufficiently early to be stored successfully within R_f. If this situation occurs, the local data path shown in Figure 50.17 fails to perform as expected and it is said that a timing *failure* or *violation* has been created. This form of timing violation is typically called a *setup* (or *long path*) violation. A setup violation is depicted in Figure 50.18 and is used in the following discussion.

The identical clock period of the clock signals C_i and C_f is shaded for identification in Figure 50.18. Also shaded in Figure 50.18 are those portions of the data signals D_i, Q_i, and D_f that are relevant to the operation of the local data path shown in Figure 50.17. Specifically, the shaded portion of D_i corresponds to the data stored in R_i at the beginning of the kth clock period. This data signal propagates to the output of the register R_i and is illustrated by the shaded portion of Q_i shown in Figure 50.18. The combinational logic operates on Q_i during the kth clock period. The result of this operation is the shaded portion of the signal D_f which must be stored in R_f during the next $(k+1)$th clock period.

Observe that as illustrated in Figure 50.18, the leading edge of C_i that initiates the kth clock period occurs at time $t_{cd}^{i} + kT_{CP}$. Similarly, the leading edge of C_f that initiates the $(k+1)$th clock period occurs at time $t_{cd}^{f} + (k+1)T_{CP}$. Therefore, the *latest arrival time* t_{AM}^{Ff} of D_f at R_f must satisfy

$$t_{AM}^{Ff} \leq [t_{cd}^{f} + (k+1)T_{CP} - \Delta_L^F] - \delta_S^{Ff} \tag{50.10}$$

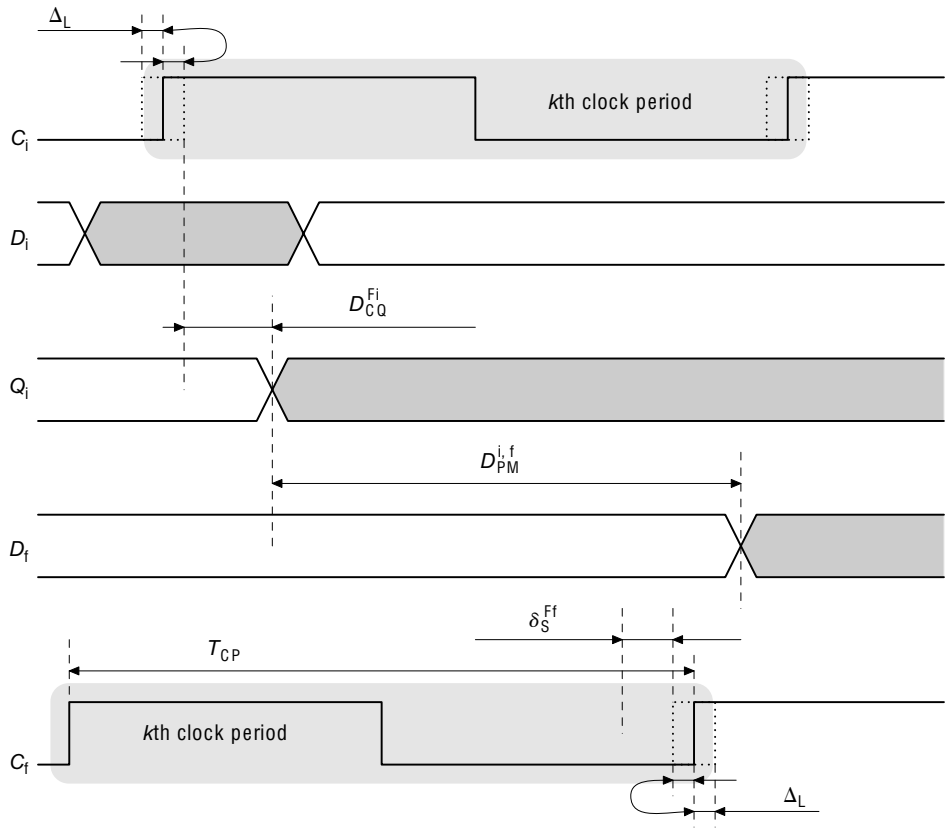


FIGURE 50.18 Timing diagram of a local data path with flip-flops with violation of the setup constraint.

The term $[t_{cd}^{f} + (k+1)T_{CP} - \Delta_L^F]$ on the right-hand side of Eq. (50.10) corresponds to the critical situation of the leading edge of C_f arriving earlier by the maximum possible deviation Δ_L^F. The $-\delta_S^{Ff}$ term on the right-hand side of Eq. (50.10) accounts for the setup time of R_f (recall the definition of δ_S^F from Section 50.4.5.3). Note that the value of t_{AM}^{Ff} in Eq. (50.10) consists of two components:

1. The latest arrival time t_{QM}^{Fi} that a valid data signal Q_i appears at the output of R_i, that is, the sum $t_{QM}^{Fi} = t_{cd}^{i} + kT_{CP} + \Delta_L^F + D_{CQM}^{Fi}$ of the latest possible arrival time of the leading edge of C_i and the maximum clock-to-Q delay of R_i.
2. The maximum propagation delay $D_{PM}^{i,f}$ of the data signals through the combinational logic block L_{if} and interconnect along the path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$.

Therefore, t_{AM}^{Ff} can be described as

$$t_{AM}^{Ff} = t_{QM}^{Fi} + D_{PM}^{i,f} = \left(t_{cd}^{i} + kT_{CP} + \Delta_L^F + D_{CQM}^{Fi}\right) + D_{PM}^{i,f} \tag{50.11}$$

By substituting Eq. (50.11) into Eq. (50.10), the timing condition guaranteeing correct signal arrival at the data input D of R_f is

$$(t_{cd}^{i} + kT_{CP} + \Delta_L^F + D_{CQM}^{Fi}) + D_{PM}^{i,f} \le \left[t_{cd}^{f} + (k+1)T_{CP} - \Delta_L^F\right] - \delta_S^{Ff} \tag{50.12}$$

The above inequality can be transformed by subtracting the kT_{CP} terms from both sides of Eq. (50.12). Furthermore, certain terms in Eq. (50.12) can be grouped together and, by noting that $t_{cd}^{i} - t_{cd}^{f} = T_{Skew}(i,f)$ is the clock skew between the registers R_i and R_f,

$$T_{Skew}(i,f) + 2\Delta_L^F \le T_{CP} - \left(D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff}\right) \tag{50.13}$$

Note that a violation of Eq. (50.13) is illustrated in Figure 50.18.

The timing relationship Eq. (50.13) represents three important results describing the late arrival of the signal D_f at the data input of the final register R_f in a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$:

1. Given *any* value of $T_{Skew}(i,f), \Delta_L^F, D_{PM}^{i,f}, \delta_S^{Ff}$, and D_{CQM}^{Fi}, the late arrival of the data signal at R_f can be prevented by controlling the value of the clock period T_{CP}. A sufficiently large value of T_{CP} can always be chosen to relax Eq. (50.13) by increasing the upper bound described by the right-hand side of Eq. (50.13).
2. For correct operation, the clock period T_{CP} does *not* necessarily have to be greater than the term $(D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff})$. If the clock skew $T_{Skew}(i,f)$ is properly controlled, choosing a particular negative value[*§] for the clock skew will relax the left side of Eq. (50.13), thereby permitting (13) to be satisfied despite $T_{CP} - (D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff}) < 0$.
3. Both the term $2\Delta_L^F$ and the term $(D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff})$ are harmful in the sense that these terms impose a *lower bound* on the clock period T_{CP} (as expected). Although negative skew can be used to relax the inequality Eq. (50.13), these two terms work against relaxing the values of T_{CP} and $T_{Skew}(i,f)$.

Finally, the relationship Eq. (50.13) may be rewritten in a form that clarifies the upper bound on the clock skew $T_{Skew}(i,f)$ imposed by Eq. (50.13):

$$T_{Skew}(i,f) \le T_{CP} - (D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff}) - 2\Delta_L^F \tag{50.14}$$

[*§] More precisely, any negative value within a specified range.

50.4.7.2 Preventing the Early Arrival of the Data Signal in a Local Data Path with Flip-Flops

Late arrival of the signal D_f at the data input of R_f (see Figure 50.17) is analyzed in Section 50.4.7.1. In this section, the analysis of the timing relationships of the local data path $\mathsf{R}_\mathrm{i} \rightsquigarrow \mathsf{R}_\mathrm{f}$ to prevent early data arrival of D_f is presented. To this end, recall from the discussion in Section 50.4.5.4 that any data signal D_f stored in R_f must lag the arrival of the leading edge of C_f by at least δ_H^{Ff}. It is possible for the opposite event to occur, that is, for a new data D_f^{new} to overwrite the value of D_f and be stored within the register R_f. If this situation occurs, the local data path shown in Figure 50.17 will not perform as desired because of a catastrophic timing violation known as a *hold* (or *short path*) violation.

In this section, hold timing violations are analyzed. It is shown that a hold violation is more dangerous than a setup violation since a hold violation cannot be removed by simply adjusting the clock period T_{CP} (unlike the case of a data signal arriving late where T_{CP} can be increased to satisfy Eq. [50.13]). A hold violation is depicted in Figure 50.19.

Note that in Figure 50.18, a data signal stored in R_i during the kth clock period arrives too late to be stored in R_f during the $(k+1)$th clock period. In Figure 50.19, however, the data stored in R_i during the kth clock period arrives at R_f too early and *destroys* the data stored in R_f during the same kth clock period. To clarify this concept, certain portions of the data signals in Figure 50.19 are shaded for easy identification. The data D_i stored in R_i at the beginning of the kth clock period is shaded. This data signal propagates to the output of the register R_i and is illustrated by the shaded portion of Q_i shown

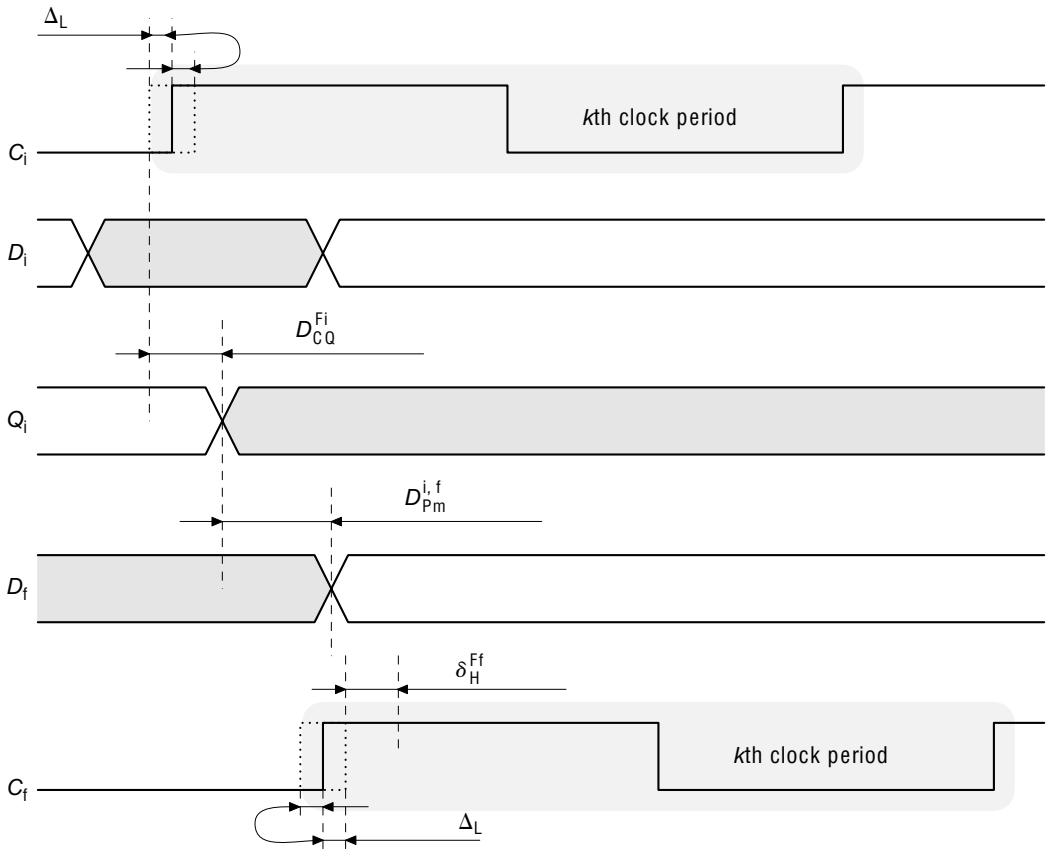


FIGURE 50.19 Timing diagram of a local data path with flip-flops featuring a hold time violation.

in Figure 50.19. The output of the logic (left unshaded part in Figure 50.19) is stored within the register R_{f} at the beginning of the $(k+1)$th clock period. Finally, the shaded portion of D_f corresponds to the data that must be stored in R_{f} at the beginning of the kth clock period.

Note that, as illustrated in Figure 50.19, the leading (or latching) edge of C_i that initiates the kth clock period occurs at time $t_{cd}^{\mathrm{i}} + kT_{CP}$. Similarly, the leading (or latching) edge of C_f that initiates the kth clock period occurs at time $t_{cd}^{\mathrm{f}} + kT_{CP}$. Therefore, the earliest arrival time t_{Am}^{Ff} of the data signal D_f at the register R_{f} must satisfy the following condition:

$$t_{Am}^{Ff} \geq (t_{cd}^{\mathrm{f}} + kT_{CP} + \Delta_L^F) + \delta_H^{Ff} \tag{50.15}$$

The term $(t_{cd}^{\mathrm{f}} + kT_{CP} + \Delta_L^F)$ on the right-hand side of Eq. (50.15) corresponds to the critical situation of the leading edge of the kth clock period of C_f arriving late by the maximum possible deviation Δ_L^F. Note that the value of t_{Am}^{Ff} in Eq. (50.15) has two components.

1. The earliest arrival time t_{Qm}^{Fi} that a valid data signal Q_i appears at the output of R_{i}, that is, the sum $t_{Qm}^{Fi} = t_{cd}^{\mathrm{i}} + kT_{CP} - \Delta_L^F + D_{CQm}^{Fi}$ of the earliest arrival time of the leading edge of C_i and the minimum clock-to-Q delay of R_{i}
2. The minimum propagation delay $D_{Pm}^{\mathrm{i,f}}$ of the signals through the combinational logic block L_{if} and interconnect wires along the path $\mathsf{R}_{\mathrm{i}} \rightsquigarrow \mathsf{R}_{\mathrm{f}}$.

Therefore, t_{Am}^{Ff} can be described as

$$t_{Am}^{Ff} = t_{Qm}^{Fi} + D_{Pm}^{\mathrm{i,f}} = (t_{cd}^{\mathrm{i}} + kT_{CP} - \Delta_L^F + D_{CQm}^{Fi}) + D_{Pm}^{\mathrm{i,f}} \tag{50.16}$$

By substituting Eq. (50.16) into Eq. (50.15), the timing condition that guarantees that D_f does *not* arrive too early at R_{f} is

$$(t_{cd}^{\mathrm{i}} + kT_{CP} - \Delta_L^F + D_{CQm}^{Fi}) + D_{Pm}^{\mathrm{i,f}} \geq (t_{cd}^{\mathrm{f}} + kT_{CP} - \Delta_L^F) + \delta_H^{Ff} \tag{50.17}$$

The inequality Eq. (50.17) can be further simplified by regrouping terms and noting that $t_{cd}^{\mathrm{i}} - t_{cd}^{\mathrm{f}} = T_{\mathrm{Skew}}(i,f)$ is the clock skew between the registers R_{i} and R_{f}.

$$T_{\mathrm{Skew}}(i,f) - 2\Delta_L^F \geq -(D_{CQm}^{Fi} + D_{Pm}^{\mathrm{i,f}}) + \delta_H^{Ff} \tag{50.18}$$

Recall that a violation of Eq. (50.18) is illustrated in Figure 50.19.

The timing relationship described by Eq. (50.18) provides certain important characteristics describing the early arrival of the signal D_f at the data input of the final register R_{f} of a local data path:

1. Unlike Eq. (50.13), the inequality Eq. (50.18) does not depend on the clock period T_{CP}. Therefore, a violation of Eq. (50.18) *cannot* be corrected by simply manipulating the value of T_{CP}. A synchronous digital system with hold violations is nonfunctional, while a system with setup violations will still operate correctly at a reduced speed.[*¶] Owing to this behavior, hold violations result in catastrophic timing failure and are considered significantly more dangerous than the setup violations described in Section 50.4.7.1. In conventional zero-clock skew systems, hold violations are typically fixed by inserting delays along the local data paths. For systems where clock skew is manipulated for improved circuit performance (nonzero clock skew circuits), this method is not appropriate.[‖*]

[*¶]Increasing the clock period T_{CP} to satisfy Eq. (50.13) is equivalent to reducing the frequency of the clock signal.

[*‖]A delay insertion method can be used on nonzero clock skew circuits as will be discussed in Section 50.4.9.3, but the timing characteristics are different.

2. The relationship Eq. (50.18) can be satisfied with a sufficiently large value of clock skew $T_{\text{Skew}}(i, f)$. However, both of the terms $2\Delta_L^F$ and δ_H^{Ff} are harmful in the sense that these terms impose a lower bound on the clock skew $T_{\text{Skew}}(i, f)$. Although positive skew may be used to relax Eq. (50.18), these two terms work against relaxing the values of $T_{\text{Skew}}(i, f)$ and $(D_{CQm}^{Fi} + D_{Pm}^{i,f})$.

Finally, the relationship Eq. (50.18) can be rewritten to stress the lower bound imposed on the clock skew $T_{\text{Skew}}(i, f)$ by Eq. (50.18):

$$T_{\text{Skew}}(i, f) \geq -(D_{Pm}^{i,f} + D_{CQm}^{Fi}) + \delta_H^{Ff} + 2\Delta_L^F \tag{50.19}$$

50.4.7.3 Clock Skew Scheduling of Edge-Triggered Circuits

Previous research [20,26,37] has indicated that tight control over the clock skew rather than the clock delays is necessary for the circuit to operate reliably. Relationships Eq. (50.14) and Eq. (50.19) are used in Ref. [37] to determine a *permissible range* of the allowed clock skew for each local data path of a circuit with edge-triggered flip-flops. The concept of a permissible range for the clock skew $T_{\text{Skew}}(i, f)$ of a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ is illustrated in Figure 50.20. For simplicity, the tolerances Δ_L^F and Δ_T^F of the clock signal are ignored in the formulation. When $T_{\text{Skew}}(i, f) \in [(-D_{Pm}^{i,f} - D_{CQm}^{Fi} + \delta_H^{Ff}), (T_{CP} - D_{PM}^{i,f} - D_{CQM}^{Fi} + \delta_S^F)]$—as shown in Figure 50.20—Eq. (50.14) and Eq. (50.19) are satisfied. The clock skew $T_{\text{Skew}}(i,f)$ is not *permitted* to be in the interval $(-\infty, -D_{Pm}^{i,f} - D_{CQm}^{Fi} + \delta_H^{Ff})$ because a race condition will be created. The clock skew is not *permitted* to be in the interval $(T_{CP} - D_{PM}^{i,f} - D_{CQM}^{Fi} - \delta_S^{Ff}, +\infty)$, either, because in that case, the minimum clock period will be limited.

Also, note that the reliability of a circuit is related to how well the circuit is protected against potential timing violations. Therefore, the reliability of any local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ of a circuit (and therefore of the entire circuit) is increased in two ways:

1. By choosing the clock skew $T_{\text{Skew}}(i, f)$ for a local data path as far as possible from the borders of the permissible range interval, that is, by (ideally) positioning the clock skew $T_{\text{Skew}}(i, f)$ in the middle of the permissible range:

$$T_{\text{Skew}}(i, f) = \frac{1}{2}[T_{CP} - (D_{PM}^{i,f} + D_{Pm}^{i,f}) - (D_{CQm}^{Fi} + D_{CQM}^{Fi} + \delta_S^{Ff} - \delta_H^{Ff})]$$

2. By increasing the width of the permissible range of the local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$.

Owing to the linear dependence of the clock skews shown in Section 50.3.4.1, however, it may *not* be possible to build a typical circuit such that for each local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$, the clock skew $T_{\text{Skew}}(i, f)$ is in the middle of the permissible range. Alternative methods are possible to solve this problem as close as possible to the *ideal* solution [20].

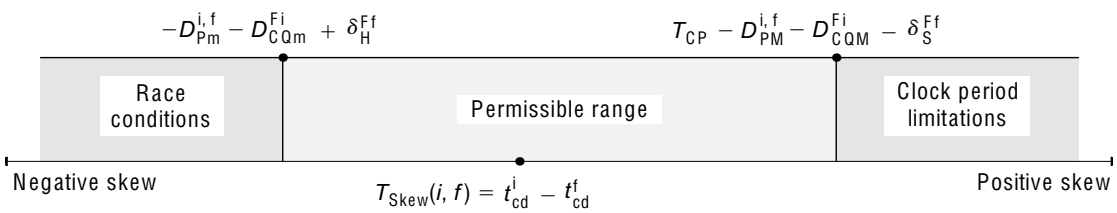


FIGURE 50.20 The permissible range of the clock skew of a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$. A timing violation exists if $T_{\text{Skew}}(i, f) \notin [(-D_{Pm}^{i,f} - D_{CQm}^{Fi} + \delta_H^{Ff}), (T_{CP} - D_{PM}^{i,f} - D_{CQM}^{Fi} - \delta_S^{Ff})]$.

TABLE 50.3 LP Model Clock Skew Scheduling of Edge-Triggered Circuits Targeting Minimum Clock Period

$$\begin{aligned} \min \quad & T_{CP} \\ \text{s.t.} \quad & T_{\text{Skew}}(i,f) \le T_{CP} - D_{PM}^{\text{i,f}} - D_{CQM}^{\text{Fi}} - \delta_S^{\text{Ff}} \\ & T_{\text{Skew}}(i,f) \ge -D_{Pm}^{\text{i,f}} - D_{CQm}^{\text{Fi}} + \delta_H^{\text{Ff}} \end{aligned}$$

TABLE 50.4 QP Model Clock Skew Scheduling of Edge-Triggered Circuits Targeting Safety Against Process Parameter Variations

$$\begin{aligned} \min \quad & \epsilon = (s-g)^2 = \textstyle\sum_{k=1}^{p}(s_k - g_k)^2 \\ \text{s.t.} \quad & Bs = 0 \\ & l_k \le s_k \le u_k \ \text{ for } k \in \{1 \cdots p\} \end{aligned}$$

Fishburn first demonstrated in Ref. [34] how linear programming techniques can be used to solve for a nontrivial clock schedule T_{CD} so as to satisfy Eq. (50.14) and Eq. (50.19) while minimizing the clock period T_{CP}. This linear programming formulation is presented in Table 50.3. Moreover, Fishburn used his LP framework to formulate the clock skew scheduling problem such that the timing reliability of a circuit is improved.

Since Fishburn's pioneering studies, the clock skew scheduling problem of edge-triggered circuits has been extensively addressed by linear and quadratic programming approaches [20,34,60–62]. Most effective of these approaches reported to date is the quadratic programming approach described in Ref. [20], which is given in Table 50.4. In Table 50.4, s are g are vectors of actual and target skews for the local data paths. From a reliability perspective, the target skew g_k for each local data path p_k can be identified as the midpoint of the path dependent permissible ranges. Mathematically, the midpoint is computed $g_k = (l_k + u_k)/2$, where l_k and u_k are the lower and upper bounds of the permissible range illustrated in Figure 50.20. The objective of this QP formulation is to minimize the least square error (LSE) of the actual skew values over the target skew values. The problem constraints are a reiteration of the permissible range requirements illustrated in Figure 50.20 and the linear dependency properties of the clock skew values discussed in Section 50.3.4.1.

Overall, the aggregate of experimental results reported in [20,34,60–63] suggest that clock skew scheduling of edge-triggered circuits permits *approximately* 30% shorter clock periods on average as compared to conventional, zero clock skew, edge-triggered circuits.[†*] Most popular solutions exhibit run times comparable to the run times of conventional timing analysis methods of zero clock skew circuits. The scalability of these clock skew scheduling methods, however, are not highly comparable to those of conventional timing analysis methods because of the necessity to simultaneously analyze all (nonzero clock skew) timing paths in a typical clock skew scheduling application.

50.4.8 Analysis of a Single-Phase Local Data Path with Latches

A local data path consisting of two level-sensitive registers (or latches) and the combinational logic between these registers (or latches) is shown in Figure 50.21. Similar to the analysis in Section 50.4.7, a single-phase synchronization clock is selected. An analysis of the timing properties of the local data path shown in Figure 50.21 is offered in the following sections. The timing relationships to prevent the late arrival of the data signal at the latch R_f are examined in Section 50.4.8.1. The timing relationships to prevent the early arrival of the data signal at the latch R_f are examined in Section 50.4.8.2.

[†*] The experimental results are reported for clock period minimization of the ISCAS'89 suite of benchmark circuits.

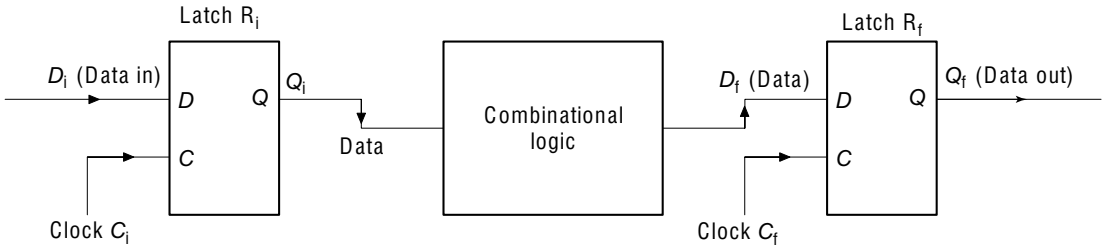


FIGURE 50.21 A single-phase local data path with latches.

The analyses presented in this section build on assumptions regarding the timing relationships among the signals of a latch similar to those assumptions used in Section 50.4.7. Specifically, it is guaranteed that every data signal arrives at the data input of a latch no later than δ_S^L time before the trailing clock edge. Also, this data signal must remain stable at least δ_H^L time after the trailing edge, that is, no new data signal should arrive at a latch δ_H^L time after the latch has become opaque.

Note that these operational properties of latches are not identical to flip-flops. In flip-flops, the setup and hold requirements described above are relative to the *leading*—not to the trailing—edge of the clock signal. This behavior is due to the transparency of the latches during the active level of the clock signal. This transparency permits clock periods smaller than the largest data propagation time, even on a zero clock skew circuit (remember Eq. [50.4]), by arriving after the leading edge of the clock signal and before the trailing edge:

$$\min T_{CP} \leq \max_{\forall \langle \mathsf{R_i}, \mathsf{R_f} \rangle} (D_{PM}^{\mathrm{if}}) \tag{50.20}$$

This operational property of latches is called *time borrowing* [53] (or *cycle stealing* [56]), as the propagation on one local data path *borrows* (or *steals*) time from the propagation on the next local data path by arriving after the leading edge. A nonzero clock skew synchronous circuit with latches can benefit both from the positive impact of clock skew scheduling and the inherent advantageous property of time borrowing.

50.4.8.1 Preventing the Late Arrival of the Data Signal in a Local Data Path with Latches

A data signal propagation scenario similar to the example illustrated in Figure 50.18 is assumed in the following discussion. A data signal D_i is stored in the latch $\mathsf{R_i}$ during the kth clock period. The data Q_i stored in $\mathsf{R_i}$ propagates through the combinational logic $\mathsf{L_{if}}$ and the interconnect along the path $\mathsf{R_i} \rightsquigarrow \mathsf{R_f}$. In the $(k+1)$th clock period, the result D_f of the computation in $\mathsf{L_{if}}$ is stored within the latch $\mathsf{R_f}$. The signal D_f must arrive at least δ_S^L time before the trailing edge of C_f in the $(k+1)$th clock period.

Similar to the discussion presented in Section 50.4.7.1, the *latest arrival time* t_{AM}^{If} of D_f at the D input of $\mathsf{R_f}$ must satisfy

$$t_{AM}^{If} \leq [t_{cd}^{\mathrm{f}} + (k+1)\, T_{CP} + C_{Wm}^L - \Delta_T^L] - \delta_S^{If} \tag{50.21}$$

Note the difference between Eq. (50.21) and Eq. (50.10). In Eq. (50.10), the first term on the right-hand side is $[t_{cd}^{\mathrm{f}} + (k+1)\, T_{CP} - \Delta_L^F]$, while in Eq. (50.21), the first term on the right-hand side has an additional term C_{Wm}^L. The addition of C_{Wm}^L is due to the characteristic that unlike flip-flops, a data signal is stored in the latches, shown in Figure 50.21, at the trailing edge of the clock signal (the C_{Wm}^L term). Similar to the case of flip-flops in Section 50.4.7.1, the term $[t_{cd}^{\mathrm{f}} + (k+1)\, T_{CP} + C_{Wm}^L - \Delta_T^L]$ in the right-hand side of

Eq. (50.21) corresponds to the critical situation of the trailing edge of the clock signal C_f arriving earlier by the maximum possible deviation Δ_T^L.

Observe that the value of t_{AM}^{Lf} in Eq. (50.21) consists of two components:

1. The latest arrival time t_{QM}^{Li} when a valid data signal Q_i appears at the output of the latch R_i
2. The maximum signal propagation delay through the combinational logic block L_if and the interconnect along the path $\mathsf{R}_\mathsf{i} \rightsquigarrow \mathsf{R}_\mathsf{f}$

The arrival time t_{AM}^{Lf} therefore can be defined as

$$t_{AM}^{Lf} = D_{PM}^{i,f} + t_{QM}^{Li} \tag{50.22}$$

However, unlike the situation of flip-flops discussed in Section 50.4.7.1, the term t_{QM}^{Li} on the right-hand side of Eq. (50.22) is not the sum of the delays through the register R_i. This characteristic occurs because the value of t_{QM}^{Li} depends upon whether the signal D_i arrived *before* or *during* the transparent state of R_i in the kth clock period. The value of t_{QM}^{Li} in Eq. (50.22) is therefore the greater of the following two quantities:

$$t_{QM}^{Li} = \max\left[\left(t_{AM}^{Li} + D_{DQM}^{Li}\right), \left(t_{cd}^{i} + kT_{CP} + \Delta_L^L + D_{CQM}^{Li}\right)\right] \tag{50.23}$$

There are two terms in the right-hand side of Eq. (50.23):

1. The term $(t_{AM}^{Li} + D_{DQM}^{Li})$ corresponds to the situation in which D_i arrives at R_i after the leading edge of the kth clock period
2. The term $\left(t_{cd}^{i} + kT_{CP} + \Delta_L^L + D_{CQM}^{Li}\right)$ corresponds to the situation in which D_i arrives at R_i before the leading edge of the kth clock pulse arrives

By substituting Eq. (50.23) into Eq. (50.22), the latest time of arrival t_{AM}^{Lf} is

$$t_{AM}^{Lf} = D_{PM}^{i,f} + \max\left[\left(t_{AM}^{Li} + D_{DQM}^{Li}\right), \left(t_{cd}^{i} + kT_{CP} + \Delta_L^L + D_{CQM}^{Li}\right)\right] \tag{50.24}$$

which is substituted into Eq. (50.21) to obtain

$$D_{PM}^{i,f} + \max\left[\left(t_{AM}^{Li} + D_{DQM}^{Li}\right), \left(t_{cd}^{i} + kT_{CP} + \Delta_L^L + D_{CQ}^{Li}\right)\right] \le \left[t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^L - \Delta_T^L\right] - \delta_S^{Lf} \tag{50.25}$$

Eq. (50.25) is an expression for the inequality that must be satisfied to prevent the late arrival of a data signal at the data input D of the register R_f. By satisfying Eq. (50.25), setup violations in the local data path with latches shown in Figure 50.21 are avoided. For a circuit to operate correctly, Eq. (50.25) must be enforced for any local data path $\mathsf{R}_\mathsf{i} \rightsquigarrow \mathsf{R}_\mathsf{f}$ consisting of the latches R_i and R_f.

The max operation in Eq. (50.25) creates a mathematically difficult situation because it is unknown which of the quantities under the max operation is greater. To overcome this obstacle, this max operation may be split into two conditions:

$$D_{PM}^{i,f} + (t_{AM}^{Li} + D_{DQM}^{Li}) \le [t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^L + \Delta_T^L] - \delta_S^{Lf} \tag{50.26}$$

$$D_{PM}^{i,f} + \left(t_{cd}^{i} + kT_{CP} + \Delta_L^L + D_{CQM}^{Li}\right) \le \left[t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^L - \Delta_T^L\right] - \delta_S^{Lf} \tag{50.27}$$

Noting that the clock skew $T_{\text{Skew}}(i,f) = t_{cd}^{i} - t_{cd}^{f}$, Eq. (50.26) and Eq. (50.27) can be rewritten as

$$D_{PM}^{i,f} + \left(t_{AM}^{Li} + D_{DQM}^{Li}\right) \le \left[t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^L - \Delta_T^L\right] - \delta_S^{Lf} \tag{50.28}$$

$$T_{\text{Skew}}(i,f) + \left(\Delta_L^L + \Delta_T^L\right) \le \left(T_{CP} + C_{Wm}^L\right) - \left(D_{CQM}^{Li} + D_{PM}^{i,f} + \delta_S^{Lf}\right) \tag{50.29}$$

Similar to Sections 50.4.7.1 and 50.4.7.2, Eq. (50.29) can be rewritten in a form that clarifies the upper bound on the clock skew $T_{\text{Skew}}(i, f)$ imposed by Eq. (50.29):

$$D_{PM}^{i,f} + \left(t_{AM}^{Li} + D_{DQM}^{Li}\right) \le \left[t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^L - \Delta_T^L\right] - \delta_S^{Lf} \tag{50.30}$$

$$T_{\text{Skew}}(i,f) \le \left(T_{CP} + C_{Wm}^L - \Delta_L^L - \Delta_T^L\right) - \left(D_{CQM}^{Li} + D_{PM}^{i,f} + \delta_S^{Lf}\right) \tag{50.31}$$

50.4.8.2 Preventing the Early Arrival of the Data Signal in a Local Data Path with Latches

A data signal propagation scenario similar to the example illustrated in Figure 50.19 is assumed in the following discussion. Recall the difference between the late arrival of a data signal at R_f and the early arrival of a data signal at R_f (see Section 50.4.7.2). In the former case, the data signal stored in the latch R_i during the kth clock period arrives too late to be stored in the latch R_f during the $(k + 1)$th clock period. In the latter case, the data signal stored in the latch R_i during the kth clock period propagates to the latch R_f too early and overwrites the data signal that was already stored in the latch R_f during the same kth clock period. These constraints hold true for synchronous circuits with latches as well, with some changes due to the subtle differences in operation between flip-flops and latches.

In order for the proper data signal to be successfully latched within R_f during the kth clock period, there should not be any changes in the signal D_f until at least the hold time after the arrival of the storing (trailing) edge of the clock signal C_f. The earliest arrival time t_{Am}^{Lf} of the data signal D_f at the register R_f must therefore satisfy the following condition:

$$t_{Am}^{Lf} \ge \left(t_{cd}^{f} + kT_{CP} + C_{Wm}^L + \Delta_T^L\right) + \delta_H^{Lf} \tag{50.32}$$

The term $\left(t_{cd}^{f} + kT_{CP} + C_{Wm}^L + \Delta_T^L\right)$ on the right-hand side of Eq. (50.32) corresponds to the critical situation of the trailing edge of the kth clock period of the clock signal C_f arriving late by the maximum possible deviation Δ_T^L Note that the value of t_{Am}^{Lf} in Eq. (50.32) consists of two components:

1. The earliest arrival time t_{QM}^{Li} that a valid data signal Q_i appears at the output of the latch R_i, that is, the sum $t_{Qm}^{Li} = t_{cd}^{i} + kT_{CP} - \Delta_L^L + D_{CQm}^{Li}$ of the earliest arrival time of the leading edge of the clock signal C_i and the minimum clock-to-Q delay D_{CQm}^{Li} of R_f.
2. The minimum propagation delay $D_{Pm}^{i,f}$ of the signal through the combinational logic L_{if} and the interconnect along the path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$.

Therefore, t_{Am}^{Lf} can be described as

$$t_{Am}^{Lf} = t_{Qm}^{Li} + D_{Pm}^{i,f} = \left(t_{cd}^{i} + kT_{CP} - \Delta_L^L + D_{CQm}^{Li}\right) + D_{Pm}^{i,f} \tag{50.33}$$

By substituting Eq. (50.33) into Eq. (50.32), the timing condition guaranteeing that D_f does *not* arrive too early at the latch R_f is

$$\left(t_{cd}^{i} + kT_{CP} - \Delta_L^L + D_{CQm}^{Li}\right) + D_{Pm}^{i,f} \ge \left(t_{cd}^{f} + kT_{CP} + C_{Wm}^L + \Delta_T^L\right) + \delta_H^{Lf} \tag{50.34}$$

The inequality shown in Eq. (50.34) can be further simplified by reorganizing the terms and noting that $t_{cd}^{i} - t_{cd}^{f} = T_{\text{Skew}}(i, f)$ is the clock skew between the registers R_i and R_f:

$$T_{\text{Skew}}(i,f) - \left(\Delta_L^L + \Delta_T^L\right) \ge -\left(D_{CQm}^{Li} + D_{Pm}^{i,f}\right) + \delta_H^{Lf} \tag{50.35}$$

The timing relationship described by Eq. (50.35) represents two important results describing the early arrival of the signal D_f at the data input of the final latch R_f of a local data path:

1. The relationship in Eq. (50.35) does not depend on the value of the clock period T_{CP}. Therefore, if a hold timing violation in a synchronous system has occurred,[††] this timing violation is catastrophic.
2. The relationship in Eq. (50.35) can be satisfied with a sufficiently large value of the clock skew $T_{\mathrm{Skew}}(i,f)$. Furthermore, both the term $(\Delta_L^L + \Delta_T^L)$ and the term δ_H^{Lf} are harmful in the sense that these terms impose a lower bound on the clock skew $T_{\mathrm{Skew}}(i,f)$. Although positive skew $T_{\mathrm{Skew}}(i,f) > 0$ can be used to relax Eq. (50.35), these two terms make it difficult to satisfy the inequality for specific values of $T_{\mathrm{Skew}}(i,f)$ and $(D_{CQm}^{Li} - D_{Pm}^{i,f})$.

Furthermore, the relationship can be rewritten to emphasize the lower bound on the clock skew $T_{\mathrm{Skew}}(i,f)$ imposed by Eq. (50.35):

$$T_{\mathrm{Skew}}(i,f) \geq \left(\Delta_L^L + \Delta_T^L\right) - \left(D_{CQm}^{Li} + D_{Pm}^{i,f}\right) + \delta_H^{Lf} \tag{50.36}$$

50.4.8.3 Clock Skew Scheduling of Level-Sensitive Circuits

Level-sensitive circuits are gaining popularity in state-of-the-art integrated circuits due to the smaller size, lower power consumption, and higher speed operation [64–66]. A timing analysis of level-sensitive circuits, however, is difficult, as outlined in Sections 50.4.8.1 and 50.4.8.2. In particular, the transparency property of latches imposes nonlinear timing constraints such as Eq. (50.25).

In conventional timing analysis (without clock skew scheduling), the nonlinearity of the timing constraints are solved with iterative approaches [59,67,68]. The application of clock skew scheduling, however, requires a more sophisticated framework than these types of iterative processes. In Ref. [69], a linear programming approach to solve the clock skew scheduling problem of level-sensitive circuits is proposed. This LP solution, presented in Table 50.5, proposes the mechanics to linearize the originally nonlinear timing constraints. In Table 50.5, the term $FI(j)$ represents the fanin of a register R_j. The tolerances Δ_L^L and Δ_T^L of the clock signal, for simplicity, are ignored in this formulation. This LP problem formulation is used as a framework to address various timing analysis problems of synchronous circuits with latches.

As shown in Refs. [69,70], nonzero clock skew, level-sensitive circuits *might* permit improved circuit performance as compared to nonzero clock skew, edge-triggered circuits. The experimental results reported for nonzero clock skew, level-sensitive circuits indicate that the minimum clock period for this type of circuit

TABLE 50.5 LP Model of Clock Skew Scheduling of Level-Sensitive Circuits Targeting the Minimum Clock Period

$$\begin{aligned}
&\min T_{CP} + M\Big[\sum_{\forall j}(t_{Qm}^{Lj} + t_{QM}^{Lj}) + \sum_{\forall j:|FI(j)|\geq 1}(t_{AM}^{Lj} + t_{Am}^{Lj})\Big] \\
&s.t. \quad t_{Am}^{Lf} \geq \delta_H^{Lf} \\
&\quad t_{AM}^{Lf} \leq T_{CP} - \delta_S^{Lf} \\
&\quad t_{Qm}^{Li} \geq t_{Am}^{Li} + D_{DQm}^{Li} \\
&\quad t_{Qm}^{Li} \geq T_{CP} - C_W^L + D_{CQm}^{Li} \\
&\quad t_{QM}^{Li} \geq t_{AM}^{Li} + D_{DQM}^{Li} \\
&\quad t_{QM}^{Li} \geq T_{CP} - C_W^L + D_{DQM}^{Li} \\
&\quad \forall n: t_{Am}^{Lf} \leq t_{Qm}^{Li_n} + D_{Pm}^{i_n,f} + T_{\mathrm{skew}}(i_n, f) - T_{CP} \\
&\quad \forall n: t_{AM}^{Lf} \geq t_{QM}^{Li_n} + D_{PM}^{i_n,f} + T_{\mathrm{skew}}(i_n, f) - T_{CP} \\
&\quad t_{AM}^{Lf} \geq t_{Am}^{Lf} \\
&\quad t_{QM}^{Lf} \geq t_{Qm}^{Lf}
\end{aligned}$$

[††]As described by the inequality (35) not being satisfied.

is comparable to the minimum clock period observed in nonzero clock skew, edge-triggered circuits (Section 50.4.7.3)—approximately 30% shorter clock periods on average are reported for both types of circuits. However, for some circuits (from the experimental benchmark circuits), level-sensitive circuits are shown to be superior. The improved operational characteristics of these circuits are due to simultaneously considering time borrowing (due to the inherent transparency property of latches) and clock skew scheduling.

50.4.9 Limitations in System Timing

Both zero clock skew and nonzero clock skew circuits are subject to limitations in the minimum clock period at which these circuits are fully operational. Remember from Section 50.3.3 that the limit for a zero clock skew circuit is the slowest local data path of the circuit (the path with the largest data propagation time $D_{PM}^{\mathrm{i,f}}$). Consequently, a timing analysis of zero clock skew circuits is centered around identifying the N slowest local data paths of a circuit and ensuring that there are no timing hazards on any of the local data paths for a given clock schedule and a given clock period. Typically, this type of timing analysis is performed with the goal of satisfying all setup time constraints on the N selected paths. As mentioned in Sections 50.4.7 and 50.4.8, this objective can be achieved by lowering the clock frequency until all setup time constraints of the form of Eq. (50.14) [where $T_{\mathrm{Skew}}(i, f) = 0$] are satisfied. Any remaining hold time violations can then be removed by inserting delay elements—a procedure called *delay padding* [71].

The limitations on nonzero clock skew circuits are more complicated. These limitations are caused by various circuit topologies and, unlike zero clock skew circuits, both setup and hold time violations are hard to remove. The limitations on the minimum clock period of a nonzero clock skew circuits are caused by the following three factors:

1. Uncertainty of the data propagation time along the local data paths [34]
2. The total data propagation time of the data path cycles [55]
3. The difference between the total data propagation time on reconvergent paths [72]

The first of these three limitations occurs on every single local data path of a synchronous circuit while the second and third limitations only occur on those circuits where the topology of the circuit graph includes cycles and reconvergent paths, respectively. A circuit with all three limitations will ultimately be affected from the most dominant limitation. In this section, these limitations are described for edge-triggered circuits—equivalent limitations on level-sensitive circuits can be similarly derived. To simplify the presentation, it is assumed that the type of limitation that is being discussed is the most dominant.

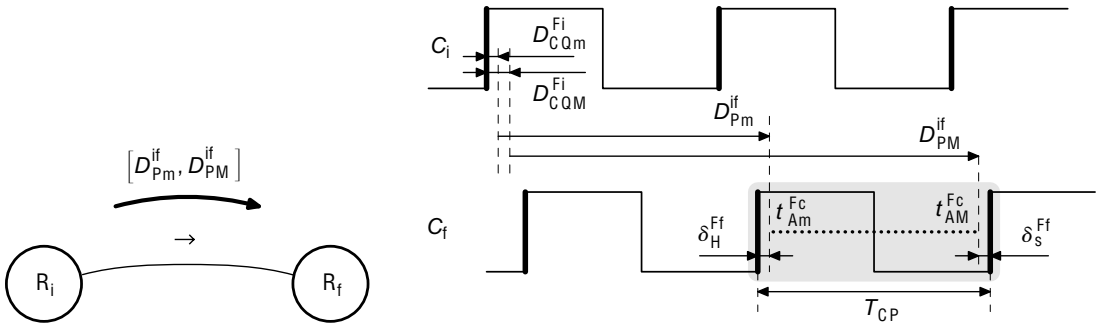
50.4.9.1 Uncertainty of Data Propagation Times

The uncertainty of the data propagation times is modeled by the min–max timing delay models (Definition 50.3.3) in timing analysis. The algebraic difference between the maximum data propagation time $D_{PM}^{\mathrm{i,f}}$ and the minimum data propagation time $D_{Pm}^{\mathrm{i,f}}$ on a local data path $\mathsf{R}_{\mathrm{i}} \rightsquigarrow \mathsf{R}_{\mathrm{f}}$ constitutes the delay uncertainty. For a critical local data path, the trailing edge of the *previous* clock cycle is the hold time before the earliest arrival of the data signal D_f at register R_{f}. The trailing edge of the *current* clock cycle is the setup time after the latest arrival of the data signal D_f at register R_{f}. This situation is depicted on an example edge-triggered local data path in Figure 50.22. Note that in Figure 50.22, the tolerance of the clock signals are ignored for the sake of simplicity. For such a critical timing path, the setup and hold time constraints [inequalities Eq. (50.10) and Eq. (50.15), respectively] satisfy the equality conditions.[††‡] Owing to this limitation, the clock period cannot be minimized any further than

$$\min T_{CP} = \max_{\forall \mathsf{R}_{\mathrm{i}} \rightsquigarrow \mathsf{R}_{\mathrm{f}}} [D_{PM}^{\mathrm{i,f}} + \delta_S^{Ff} - (D_{Pm}^{\mathrm{i,f}} + \delta_H^{Ff})] \tag{50.37}$$

The shaded region in Figure 50.22 illustrates the timing criticality, causing the limitation on T_{CP}.

[††‡] These constraints have no available slack for improvement.



(a) A sample local data path (b) Delay uncertainty in timing diagram

FIGURE 50.22 Limitation on the minimum clock period T_{CP} caused by the delay uncertainty of a local data path. (a) A sample local data path. (b) Delay uncertainty in timing diagram.

50.4.9.2 Data Path Cycles

Limitations due to data path cycles occur due to the accumulation of the timing relationships over a cycle of local data paths. In a zero clock skew circuit, the circuit topology is almost irrelevant in the timing analysis because each local data path is analyzed independent of any neighbors. The timing of local data paths of a nonzero clock skew circuit, however, is interdependent. For a cycle of local data paths, this interdependency regains the form described in Section 50.3.4.1. In this linear dependency form, the minimum clock period is further limited by the criticality of the local data paths along the cycle (in addition to the limitations caused by the delay uncertainty of each local data path along the cycle). This limitation is illustrated for a sample local data path cycle in Figure 50.23.

The cyclic traveling path for the data signal over a data path cycle, such as the example circuit shown in Figure 50.23, leads to stringent operating conditions under nonzero clock skew. The local data paths along the cycle operate without any slack time, because any existing slack on these local data paths is distributed over the paths through the mechanics of the clock skew scheduling process. In such circuits where a data path cycle is critical, the minimum clock period depends on two factors. The first factor is the number of local data paths n along the cycle. For n local data paths on the cycle, n clock cycles must have passed after each completion of the cycle on a register. The second factor is the total delay of the data signal over the local data paths along the cycle. This total delay time includes the setup time δ_S^{Ff} and maximum clock-to-output time D_{CQM}^{Fi} of each register along the cycle, the maximum data propagation time $D_{PM}^{i,f}$ of each local data path along the cycle, and the tolerances of the clock signal (which are ignored for simplicity). The limitation on the minimum clock period by the data path cycles is given by

$$\min T_{CP} = \frac{\sum_{\forall R_i \rightsquigarrow R_f \text{ on cycle}} (D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff})}{n} \tag{50.38}$$

The shaded region in Figure 50.23 illustrates the timing criticality, causing the limitation on T_{CP}.

50.4.9.3 Reconvergent Paths

A *reconvergent path* is composed of a series of two or more local data paths with a common source register (*divergent* register) and a common sink register (*convergent* register). A *reconvergent system* is composed of at least two parallel reconvergent paths. The interdependency of the timing of local data paths in a nonzero clock skew system occurs explicitly in a reconvergent system because of reconvergent fanout. A data signal that is initially stored in the divergent register starts propagating simultaneously through all of the reconvergent paths but arrives at the convergent register at (possibly) *different* times. In the case of nonidentical numbers of registers in two reconvergent paths, the data signals may arrive at the

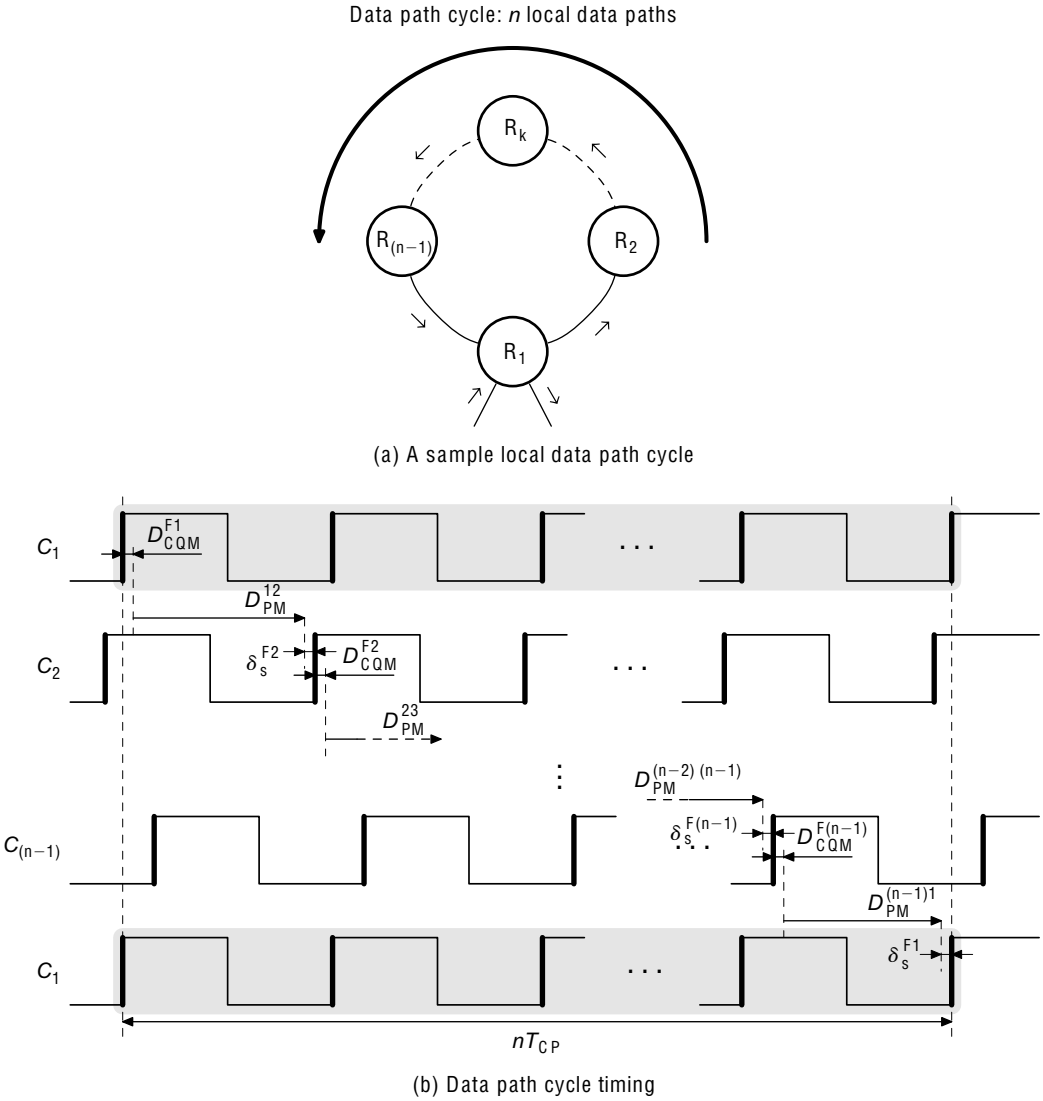


FIGURE 50.23 Limitation on the minimum clock period T_{CP} caused by data path cycles. (a) A sample local data path cycle. (b) Data path cycle timing.

convergent register during different clock cycles. The timing of all reconvergent paths is satisfied by collectively analyzing the arrival time of the data signals at the convergent register over a duration of (possibly) multiple clock cycles. In Figure 50.24, the limitation such a reconvergent system imposes on the minimum clock period of a nonzero clock skew circuit is illustrated.

In Figure 50.24, two reconvergent paths with m and n registers (excluding divergent and convergent registers) respectively, are considered. The total propagation time of the data signal on the two reconvergent paths are shown. Let the propagation time on the reconvergent paths with m and n registers be the longest and shortest total propagation times, respectively. After propagating along these two paths $(m+1)$ and $(n+1)$ clock cycles must have elapsed, respectively, by the time the data signals arrive at the convergent register. When critical, the reconvergent path with n registers is matched with the trailing edge of the nth clock cycle, while the reconvergent path with m registers is matched with the trailing edge of the $(m+1)$th clock cycle. Thus, the algebraic difference between the two total data propagation

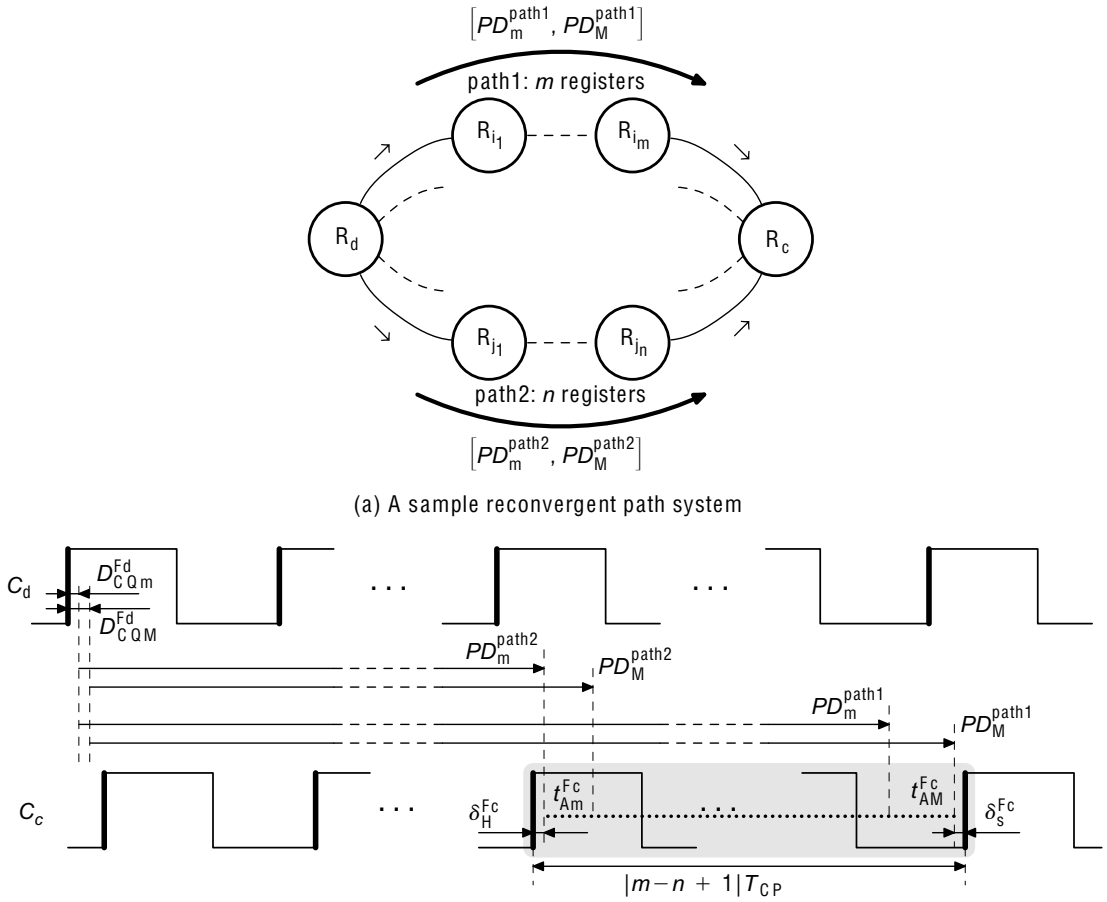


FIGURE 50.24 Limitation on the minimum clock period T_{CP} caused by reconvergent paths. (a) A sample reconvergent path system. (b) Reconvergent path system timing diagram.

times along the reconvergent paths limits the minimum clock period. Mathematically, the limitation of the reconvergent paths on the minimum clock period of nonzero clock skew circuits is given by

$$\min T_{CP} = \frac{PD_M^{\text{path1}} - PD_m^{\text{path2}} + \delta_S^{F\text{convergent}} + \delta_H^{F\text{convergent}}}{|\, m - n + 1\,|} \tag{50.39}$$

where PD_M^{pathp} and PD_m^{pathp} represent the maximum and minimum total data propagation times between the divergent and convergent registers over path 1 and path 2, respectively.

Unlike the limitations caused by the delay uncertainty of the local data paths and the total data propagation times along the data path cycles, the limitations caused by reconvergent paths can be mitigated. The mitigation procedure offered in Ref. [72] involves *systematic delay insertion* on one or more of the reconvergent paths to decrease the algebraic difference $(PD_M^{\text{path1}} - PD_m^{\text{path2}})$ of Eq. (50.39), which consequently improves the minimum clock period T_{CP}. Note that it is possible to increase the path delay PD_m^{path2} without increasing PD_M^{path1} because both paths are determined by two different series of local data paths.[†§]

[†§]The minimum and maximum total data propagation times along a reconvergent system may be observed on the same reconvergent path. In such a case, delay insertion is not beneficial.

TABLE 50.6 LP Model Clock Skew Scheduling of Edge-Triggered Circuits with Delay Insertion Method Targeting Minimum Clock Period

$$\begin{aligned}
\min \quad & T_{CP} + \sum_{\forall ? i \rightsquigarrow ? f} (I_M^{i,f} + I_m^{i,f}) \\
\text{s.t.} \quad & T_{skew}(i,f) \le T_{CP} - D_{PM}^{i,f} - D_{CQM}^{Fi} - \delta_S^{Ff} - I_M^{i,f} \\
& T_{skew}(i,f) \ge -D_{Pm}^{i,f} - D_{CQm}^{Fi} + \delta_H^{Ff} - I_m^{i,f} \\
& I_M^{i,f} \ge I_m^{i,f}
\end{aligned}$$

The systematic delay insertion method described in Ref. [72] is complicated both in theory and practice, and varies for edge- and level-sensitive circuits. The basic representation of the method presented in this section is defined on edge-triggered circuits. In the application of the delay insertion method, the timing analysis framework of Table 50.3 is used. The LP model problem formulation of the delay insertion method is presented in Table 50.6. The generated linear programming model problem provides an *automated* approach to the treatment of those limitations caused by reconvergent paths. The problem is formulated by modeling a virtual delay element on every local data path in a circuit. Normally, a refinement of the formulation is possible by modeling a delay element only on the reconvergent paths. The former formulation simply returns zero for those paths that are not reconvergent and need not be padded. The inserted delay element is modeled by the minimum $I_m^{i,f}$ and maximum $I_M^{i,f}$ delay values, agreeing with the min–max timing models used in static timing analysis.

Experimental results demonstrate that delay insertion can improve the minimum clock period of a nonzero clock skew circuit by on average approximately 10%. The actual improvement for specific circuitry and cell libraries are dependent on the practical implementation style of this characteristically rigorous, but effective, application method [72].

50.5 A Final Note and Summary

In this chapter, the general properties of system timing for synchronous circuits are outlined. The timing properties of registers and local data paths as applicable to overall system timing are analyzed. The timing hazards of synchronous circuits are defined for circuits built with both edge-triggered flip-flops and level-sensitive latches. The benefits of clock skew scheduling in improving circuit performances while eliminating timing hazards are described.

Note that in a fully synchronous digital VLSI system it is possible to encounter types of local data paths different from those circuits analyzed in this section. For example, a local data path may begin with a *positive*-polarity, edge-triggered register R_i and end with a *negative*-polarity, edge-triggered register R_f. It is also possible that different types of registers are used, e.g., a register with more than one data input, or a pulsed latch [73]. In each particular case, the analyses described in this section illustrate a general methodology to determine the proper timing relationships specific to that system. Similar reasoning can be applied to the treatment of other specific timing problems, such as clock period verification [53,59].

References

1. J.S. Kilby, Invention of the integrated circuit, *IEEE Trans. Electron Devices*, vol. ED-23, pp. 648–654, 1976.
2. G.E. Moore, Cramming more components onto integrated circuits, *Electronics*, vol. 38, 1965.
3. T. Takayanagi, J. Shin, B. Petrick, J. Su, H. Levy, H. Pham, J. Son, N. Moon, D. Bistry, M. Singh, V. Mathur, and A. Leon, A dual-core 64 b ultraSPARC microprocessor for dense server applications, *Proceedings of the IEEE International Solid-State Circuits Conference*, vol. 2, pp. 58–513, February 2004.

4. J. Clabes, J. Friedrich, M. Sweet, J. Dilullo, S. Chu, D. Plass, J. Dawson, P. Muench, L. Powell, M. Floyd, B. Sinharoy, M. Lee, M. Goulet, J. Wagoner, N. Schwartz, S. Runyon, G. Gorman, P. Restle, R. Kalla, J. McGill, and S. Dodson, Design and implementation of the POWER5™ microprocessor, *Proceedings of the IEEE International Solid-State Circuits Conference*, vol. 1, pp. 56–57, February 2004.
5. S. Naffziger, B. Stackhouse, and T. Grutkowski, The implementation of a 2-core multi-threaded Itanium-family processor, *Proceedings of the IEEE International Solid-State Circuits Conference*, pp. 182–184, February 2005.
6. D. Pham, S. Asano, M. Bolliger, M. Day, H. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, The design and implementation of a first-generation CELL processor, *Proceedings of the IEEE International Solid-State Circuits Conference*, pp. 184–186, February 2005.
7. H.B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.
8. P. Bai, C. Auth, S. Balakrishnan, M. Bost, R. Brain, V. Chikarmane, R. Heussner, M. Hussein, J. Hwang, D. Ingerly, R. James, J. Jeong, C. Kenyon, E. Lee, S.-H. Lee, N. Lindert, M. Liu, Z. Ma, T. Marieb, A. Murthy, R. Nagisetty, S. Natarajan, J. Neirynck, A. Ott, C. Parker, J. Sebastian, R. Shaheed, S. Sivakumar, J. Steigerwald, S. Tyagi, C. Weber, B. Woolery, A. Yeoh, K. Zhang, and M. Bohr, A 65nm logic technology featuring 35nm gate lengths, enhanced channel strain, 8 Cu interconnect layers, low-k ILD and 0.57 μm^2 SRAM cell, *Proceedings of the IEEE International Electron Devices Meeting*, pp. 657–660, December 2004.
9. N.W. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*. Addison-Wesley, 2nd ed., 1992.
10. J.P. Uyemura, *Introduction to VLSI Circuits and Systems*. Wiley, 2002.
11. C. Mead and L. Conway, *Introduction to VLSI Systems*. Addison-Wesley, 1980.
12. F. Anceau, A synchronous approach for clocking VLSI systems, *IEEE J. Solid-State Circuits*, vol. SC-17, pp. 51–56, 1982.
13. M. Afghani and C. Svensson, A unified clocking scheme for VLSI systems, *IEEE J. Solid State Circuits*, vol. SC-25, pp. 225–233, 1990.
14. S.H. Unger and C.-J. Tan, Clocking schemes for high-speed digital systems, *IEEE Trans. Comput.*, vol. C-35, pp. 880–895, 1986.
15. E.G. Friedman, *Clock Distribution Networks in VLSI Circuits and Systems*. IEEE Press, 1995.
16. S. Bothra, B. Rogers, M. Kellam, and C.M. Osburn, Analysis of the effects of scaling on interconnect delay in ULSI circuits, *IEEE Trans. Electron Devices*, vol. ED-40, pp. 591–597, 1993.
17. Y.I. Ismail and E.G. Friedman, Effects of inductance on the propagation delay and repeater insertion in VLSI circuits, *IEEE Trans. VLSI Syst.*, vol. 8, pp. 195–206, April 2000.
18. M. Saint-Laurent, M. Swaminathan, and J. Meindl, On the micro-architectural impact of clock distribution using multiple PLLs, *Proceedings of the IEEE International Conference on Computer Design*, pp. 214–220, September 2001.
19. J. Wood, T. Edwards, and S. Lipa, Rotary traveling-wave oscillator arrays: A new clock technology, *IEEE J. Solid-State Circuits*, vol. 36, pp. 1654–1665, 2001.
20. I.S. Kourtev and E.G. Friedman, *Timing Optimization Through Clock Skew Scheduling*. Kluwer Academic, 2000.
21. C.E. Leiserson and J.B. Saxe, A mixed-integer linear programming problem which is efficiently solvable, *J. Algorithms*, vol. 9, pp. 114–128, 1988.
22. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*. MIT Press, 1989.
23. D.B. West, *Introduction to Graph Theory*. Prentice-Hall, 1996.
24. J.L. Neves and E.G. Friedman, Topological design of clock distribution networks based on non-zero clock skew specification, *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, pp. 468–471, August 1993.
25. J.G. Xi and W.W.-M. Dai, Useful-skew clock routing with gate sizing for low power design, *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 383–388, June 1996.
26. J.L. Neves and E.G. Friedman, Design methodology for synthesizing clock distribution networks exploiting non–zero localized clock skew, *IEEE Trans. VLSI Syst.*, vol. VLSI-4, pp. 286–291, 1996.

27. M.A.B. Jackson, A. Srinivasan, and E.S. Kuh, Clock routing for high-performance ICs, *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 573–579, June 1990.
28. R.-S. Tsay, An exact zero-skew clock routing algorithm, *IEEE Trans. Computer-Aided Design Integrated Circuits Syst.*, vol. CAD-12, pp. 242–249, 1993.
29. N.-C. Chou and C.-K. Cheng, On general zero-skew clock net construction, *IEEE Trans. VLSI Systems*, vol. VLSI-3, pp. 141–146, 1995.
30. N. Ito, H. Sugiyama, and T. Konno, ChipPRISM: Clock routing and timing analysis for high-performance CMOS VLSI chips, *Fujitsu Sci. Tech. J.*, vol. 31, pp. 180–187, 1995.
31. N. Gaddis and J. Lotz, A 64-b quad-issue CMOS RISC microprocessor, *IEEE J. Solid-State Circuits*, vol. SC-31, pp. 1697–1702, 1996.
32. W.J. Bowhill et al., Circuit implementation of a 300-MHz 64-bit second-generation CMOS alpha CPU, *Digital Tech. J.*, vol. 7, pp. 100–118, 1995.
33. T.-C. Lee and J. Kong, The new line in IC design, *IEEE Spectrum*, pp. 52–58, March 1997.
34. J.P. Fishburn, Clock skew optimization, *IEEE Trans. Comput.*, vol. C–39, pp. 945–951, 1990.
35. E.G. Friedman, The application of localized clock distribution design to improving the performance of retimed sequential circuits, *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 12–17, December 1992.
36. I.S. Kourtev and E.G. Friedman, Simultaneous clock scheduling and buffered clock tree synthesis, *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1812–1815, June 1997.
37. 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.
38. L.A. Glasser and D.W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*. Addison-Wesley, 1985.
39. J.P. Uyemura, *Circuit Design for CMOS VLSI*. Kluwer Academic, 1992.
40. S.-M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*. McGraw-Hill, 1996.
41. A.S. Sedra and K.C. Smith, *Microelectronic Circuits*. Oxford University Press, 4th ed., 1997.
42. Z. Kohavi, *Switching and Finite Automata Theory*. McGraw-Hill, 2nd ed., 1978.
43. M.M. Mano and C.R. Kime, *Logic and Computer Design Fundamentals*. Prentice-Hall, Inc., 1997.
44. W. Wolf, *Modern VLSI Design: A Systems Approach*. Prentice-Hall, Inc., 1994.
45. T. Kacprzak and A. Albicki, Analysis of metastable operation in RS CMOS flip-flops, *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 57–64, 1987.
46. T.A. Jackson and A. Albicki, Analysis of metastable operation in D latches, *IEEE Trans. Circuits Syst. I Fundamental Theory Appl.*, vol. CAS I–36, pp. 1392–1404, 1989.
47. E.G. Friedman, Latching characteristics of a CMOS bistable register, *IEEE Trans. Circuits Syst. I Fundamental Theory Appl.*, vol. CAS I–40, pp. 902–908, 1993.
48. S.H. Unger, Double-edge-triggered flip-flops, *IEEE Trans. Comput.*, vol. C-30, pp. 447–451, 1981.
49. S.-L. Lu, A novel CMOS implementation of double-edge-triggered D-flip-flops, *IEEE J. Solid State Circuits*, vol. SC-25, pp. 1008–1010, 1990.
50. M. Afghani and J. Yuan, Double-edge-triggered D-flip-flops for high-speed CMOS circuits, *IEEE J. Solid State Circuits*, vol. SC-26, pp. 1168–1170, 1991.
51. R. Hossain, L. Wronski, and A. Albicki, Double edge triggered devices: Speed and power constraints, *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 1491–1494, May 1993.
52. G.M. Blair, Low-power double-edge triggered flipflop, *Electron. Lett.*, vol. 33, pp. 845–847, 1997.
53. M.R. Dagenais and N.C. Rumin, On the calculation of optimal clocking parameters in synchronous circuits with level-sensitive latches, *IEEE Trans. Computer-Aided Design*, vol. CAD-8, pp. 268–278, 1989.
54. A. Ishii, C. Leiserson, and M. Papaefthymiou, Optimizing two-phase, level-clocked circuitry, *Proceedings of the Brown/MIT Conference: Advanced Research on VLSI Parallel Systems*, pp. 245–264, March 1992.
55. M.C. Papaefthymiou and K. Randall, Edge-triggering vs. two-phase level-clocking, *Proceedings of the Symposium on Research in Integrated Systems*, pp. 201–218, March 1993.

56. I. Lin, J.A. Ludwig, and K. Eng, Analyzing cycle stealing on synchronous circuits with level-sensitive latches, *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 393–398, June 1992.
57. J. Lee, D.T. Tang, and C.K. Wong, A timing analysis algorithm for circuits with level-sensitive latches, *IEEE Trans. Computer-Aided Design*, vol. CAD-15, pp. 535–543, 1996.
58. T.G. Szymanski, Computing optimal clock schedules, *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 399–404, June 1992.
59. K.A. Sakallah, T.N. Mudge, and O.A. Olukotun, *check*T_c and *min*T_c: Timing verification and optimal clocking of synchronous digital circuits, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 552–555, November 1990.
60. C. Albrecht, B. Korte, J. Schietke, and J. Vygen, Cycle time and slack optimization for VLSI-chips, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 232–238, November 1999.
61. S. Held, B. Korte, J. Massberg, M. Ringe, and J. Vygen, Clock scheduling and clocktree construction for high performance ASICs, *Proceedings of the International Conference on Computer-Aided Design*, pp. 232–239, November 2003.
62. K. Ravindran, A. Kuehlmann, and E. Sentovich, Multi-domain clock skew scheduling, *Proceedings of the International Conference on Computer-Aided Design*, pp. 801–808, November 2003.
63. R. Mader, E.G. Friedman, A. Litman, and I.S. Kourtev, Large scale clock skew scheduling techniques for improved reliability of digital synchronous circuits, *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 357–360, May 2002.
64. S. Naffziger, G. Colon-Bonet, T. Fischer, R. Riedlinger, T. Sullivan, and T. Grutkowski, The implementation of the Itanium 2 microprocessor, *IEEE J. Solid-State Circuits*, vol. 37, pp. 1448–1460, 2002.
65. J. Warnock, Circuit design issues for the POWER4 chip, *Proceedings of the International Symposium on VLSI Technology, Systems, and Applications*, pp. 125–128, October 2003.
66. C. Webb, C. Anderson, L. Sigal, K. Shepard, J. Liptay, J.D. Warnock, B. Curran, B. Krumm, M. Mayo, P. Camporese, E. Schwarz, M. Farrell, P. Restle, R. Averill III, T. Slegel, W. Houtt, Y. Chan, B. Wile, T. Nguyen, P. Emma, D. Beece, C. Ching-Te, and C. Price, A 400-MHz S/390 microprocessor, *IEEE J. Solid-State Circuits*, vol. 32, pp. 1665–1675, 1997.
67. T.M. Burks, K.A. Sakallah, and T.N. Mudge, Critical paths in circuits with level-sensitive latches, *IEEE Trans. VLSI Syst.*, vol. 3, pp. 273–291, 1995.
68. T.G. Syzmanski and N. Shenoy, Verifying clock schedules, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 124–131, November 1992.
69. B. Taskin and I.S. Kourtev, Linear timing analysis of SOC synchronous circuits with level-sensitive latches, *Proceedings of the IEEE ASIC/SOC Conference*, pp. 358–362, September 2002.
70. B. Taskin and I.S. Kourtev, Linearization of the timing analysis and optimization of level-sensitive digital synchronous circuits, *IEEE Trans. VLSI Syst.*, vol. 12, pp. 12–27, 2004.
71. N. Shenoy, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, Minimum padding to satisfy short path constaints, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 156–161, November 1993.
72. B. Taskin and I.S. Kourtev, Delay insertion in clock skew scheduling, *Proceedings of the ACM International Symposium on Physical Design*, pp. 47–54, April 2005.
73. D. Harris, *Skew-Tolerant Circuit Design*. Morgan Kaufmann, 2001.

Appendix

Glossary of Terms

The following notations are used in this section:

1. *Clock Signal Parameters*

   T_{CP}: the clock period of a circuit

   Δ_L: the tolerance of the leading edge of any clock signal

   Δ_T: the tolerance of the trailing edge of any clock signal

   Δ_L^L: the tolerance of the leading edge of a clock signal driving a latch

   Δ_T^L: the tolerance of the trailing edge of a clock signal driving a latch

   Δ_L^F: the tolerance of the leading edge of a clock signal driving a flip-flop

   Δ_T^F: the tolerance of the trailing edge of a clock signal driving a flip-flop

   C_{Wm}^L: the minimum width of the clock signal in a circuit with latches

   C_{Wm}^F: the minimum width of the clock signal in a circuit with flip-flops

   C^{p_i}: the clock signal phase p_i

   ϕ^{p_i}: the delay of clock signal C^{p_i} with respect to common clock cycle

   t_{cd}^{i}: the delay of single-phase clock signal phase at register R_i with respect to common time reference

   t_{cd}^{i,p_i}: the delay of clock signal phase p_i at register R_i with respect to common time reference

2. *Latch Parameters*

   D_{CQ}^L: the clock-to-output delay of a latch

   D_{CQ}^{Li}: the clock-to-output delay of the latch R_i

   D_{CQm}^L: the minimum clock-to-output delay of a latch

   D_{CQm}^{Li}: the minimum clock-to-output delay of the latch R_i

   D_{CQM}^L: the maximum clock-to-output delay of a latch

   D_{CQM}^{Li}: the maximum clock-to-output delay of the latch R_i

   D_{DQ}^L: the data-to-output delay of a latch

   D_{DQ}^{Li}: the data-to-output delay of the latch R_i

   D_{DQm}^L: the minimum data-to-output delay of a latch

   D_{DQm}^{Li}: the maximum data-to-output delay of the latch R_i

   D_{DQM}^L: the minimum data-to-output delay of a latch

   D_{DQM}^{Li}: the maximum data-to-output delay of the latch R_i

   δ_S^L: the setup time of a latch

   δ_S^{Li}: the setup time of the latch R_i

   δ_H^L: the hold time of a latch

   δ_H^{Li}: the hold time of the latch R_i

   t_{AM}^L: the latest arrival time of the data signal at the data input of a latch

t_{AM}^{Li}: the latest arrival time of the data signal at the data input of the latch R_i

t_{Am}^{L}: the earliest arrival time of the data signal at the data input of a latch

t_{Am}^{Li}: the earliest arrival time of the data signal at the data input of the latch R_i

t_{QM}^{L}: the latest arrival time of the data signal at the data output of the latch

t_{QM}^{Li}: the latest arrival time of the data signal at the data output of the latch R_i

t_{Qm}^{L}: the earliest arrival time of the data signal at the data output of a latch

t_{Qm}^{Li}: the earliest arrival time of the data signal at the data output of the latch R_i

3. *Flip-Flop Parameters*

D_{CQ}^{F}: the clock-to-output delay of a latch

D_{CQ}^{Fi}: the clock-to-output delay of the latch R_i

D_{CQm}^{F}: the minimum clock-to-output delay of a latch

D_{CQm}^{Fi}: the minimum clock-to-output delay of the latch R_i

D_{CQM}^{F}: the maximum clock-to-output delay of a latch

D_{CQM}^{Fi}: the maximum clock-to-output delay of the latch R_i

δ_S^F: the setup time of a latch

δ_S^{Fi}: the setup time of the latch R_i

δ_H^F: the hold time of a latch

δ_H^{Fi}: the hold time of the latch R_i

t_{AM}^{F}: the latest arrival time of the data signal at the data input of a latch

t_{AM}^{Fi}: the latest arrival time of the data signal at the data input of the latch R_i

t_{Am}^{F}: the earliest arrival time of the data signal at the data input of a latch

t_{Am}^{Fi}: the earliest arrival time of the data signal at the data input of the latch R_i

t_{QM}^{F}: the latest arrival time of the data signal at the data output of a latch

t_{QM}^{Fi}: the latest arrival time of the data signal at the data output of the latch R_i

t_{Qm}^{F}: the earliest arrival time of the data signal at the data output of a latch

t_{Qm}^{Fi}: the earliest arrival time of the data signal at the data output of the latch R_i

4. *Local Data Path Parameters*

$R_i \rightsquigarrow R_f$: a local data path from register R_i to register R_f exists

$R_i \not\rightsquigarrow R_f$: a local data path from register R_i to register R_f does not exist