CLOCK SYSTEM DESIGN

KENNETH D. WAGNER
IBM Corp.

A well-designed clock system is a fundamental requirement in high-speed computers. In this tutorial, the author provides a framework for understanding system timing and then describes how the clock system executes the timing specifications. The tutorial examines clock generation and the construction of clock-distribution networks, which are integral to any clock system. Examples from contemporary high-speed systems highlight several common methods of clock generation, distribution, and tuning. Tight control of system clock skew is essential to an effective clock system.

The careful design of clock systems is often neglected. Part of the reason is that older, slower computers had higher tolerances to variations in the clock signal and had less exacting timing requirements. Today, however, as the demand for high-speed computers grows, the design of their clock systems should become a major concern not only in achieving high performance, but also in reducing assembly and maintenance costs.

A well-planned and well-built clock system is a prerequisite to reliable long-term computer operation. Conversely, a badly designed clock system can plague a computer throughout its lifetime, affecting its operation at any speed. To make such systems function, components often have to be tuned individually at several stages of manufacturing.

Despite these costs and performance penalties, timing design is still overlooked in many systems. Although significant decisions that must be made early in computer design include such issues as clocking scheme and type of memory element, designers seldom participate. Instead, system architects may simply repeat a previously successful set of choices, despite significant changes in design specifications, technology, and environment. Of course, these systems will eventually be functional, but they will require much more maintenance and tuning—costs not always reflected back to the developers.

These attitudes prevail in part because timing design problems are rarely reported in the literature. Also, design teams tend to be secretive about their clock systems, either because they believe they are doing something new or because they are doing nothing new and are afraid to be associated with an older technique. Either way, the result is a scarcity of information on how to avoid timing problems through proper design of the clock system.

THE CLOCK SYSTEM

System timing specifications are executed using a clock system. The clock system has two main functions, clock generation and clock distribution. We use clock-generation circuitry to form highly accurate timing signals, which we then use to synchronize

0740-7475/88/1000-09$1.00©1988 IEEE

Two types of clocked bistable elements are important in contemporary high-speed computers: the latch and the edge-triggered flip-flop.

changes in the system state. These pulsed, synchronizing signals are known as clocks. We use clock distribution to deliver the clocks to their destinations at precisely specified instants. A network, called the *clock-distribution network*, propagates clocks formed by clock generation to clocked memory elements.

Most logic design texts, such as that by McCluskey (see "Additional Reading" at the end of this article), describe bistable elesynchronized by the clock signals. A system oscillator is the source for these periodic signals. We generate and manipulate the clock signals and precisely place clock pulses to meet the system timing requirements. We may also tune the clocks to compensate for inaccuracies in the clock pulsewidth or pulse position.

BISTABLE ELEMENTS

The focus of this article is on the timing design of systems that use static bistable elements. The techniques described can also be used in the timing design for other types of clocked memory elements, such as arrays and dynamic latches, or for precharging circuitry.

Most logic design texts, such as that by McCluskey (see "Additional Reading" at the end of this article), describe bistable elements and their characteristics in great detail. Two types of clocked bistable elements are important in contemporary high-speed computers: the latch and the edge-triggered flip-flop. The latch is transparent while its clock (control) input is active. By transparent, we mean that its outputs reflect any of its data inputs. Edge-triggered elements, such as the D flip-flop, respond to their data inputs only at either the rising or falling transition of their clock input. They do not have the transparency property of the latch.

We can describe the time-dependent behavior of a bistable element using the following parameters:

- *setup time*, the minimum time that the data input of the bistable element must be held stable before the active edge or latching level of the clock pulse occurs
- *hold time*, the minimum time that the data input of the bistable element must be held stable after the active edge or latching level of the clock pulse disappears
- *propagation delay*, the time between a change on the clock or data input of the bistable element and the corresponding change on its output

For system operation to be correct, the setup time, hold time, and minimum clock pulsewidth must be satisfied for each bistable element. Signals whose propagation delay is so long that it violates the setup time are called long-path signals. Signals whose propagation delay is so short that it violates the hold time are called short-path signals. Both conditions result in incorrect data being stored.

SYSTEM CLOCKING SCHEMES

System clocking is either single-phase, multiphase (usually two-phase), or edge-triggered. Figure 1 illustrates. The dark rectangles in the figure represent the interval during which a bistable element samples its data input. Each scheme requires a minimum clock pulsewidth.

The most widely used scheme is multiphase clocking. The multiphase clocking scheme in Figure 1b is two-phase, nonoverlapping. In this scheme, two distinct clock phases are distributed within the system, and each bistable element receives one of these two clocks. Systems that have adopted two-phase clocking include microprocessors such as the Intel 80x86 series and Motorola MC68000 family, micro-mainframes such as the HP-9000, and mainframes such as the IBM 3090 and the Univac 1100/90.

Figure 2 shows a finite-state machine, a machine that realizes sequential logic functions, with each clocking scheme. (For more on finite-state machines, see McCluskey's text.) For simplicity, primary I/O is not shown. The Amdahl 580 mainframe and Cray-1 vector processor are single-phase latch machines, such as that shown in Figure 2a. Modern high-speed microprocessors like the Bellmac-32A are two-phase latch machines with a single-latch design using nonoverlapping clock phases, such as that shown in Figure 2b. Figure 2c shows a two-phase latch machine with a double-latch design. This type of machine supports scan-path testing, since it can use LSSD latch pairs, which are hazard-free master-slave latches with a scan input port. Most contemporary IBM products, including IBM 3090 mainframes, incorporate design for testability using this structure. Systems built with catalog parts are usually flip-flop machines, such as that shown in Figure 2d, because clocked bistable elements commonly offered in bipolar and CMOS MSI chips are edge-triggered.

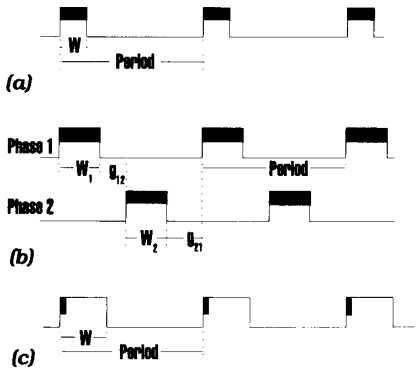

Figure 1. *System clocking waveforms; single-phase* ***(a)****, two-phase* ***(b)****, and edge-triggered* ***(c)****. W_i=pulsewidth of phase j and g_{ij}= interphase gap from phase i to phase j; if $g_{ij} > 0 \Rightarrow$ two-phase, nonoverlapping, if $g_{ij} < 0 \Rightarrow$ two-phase overlapping.*

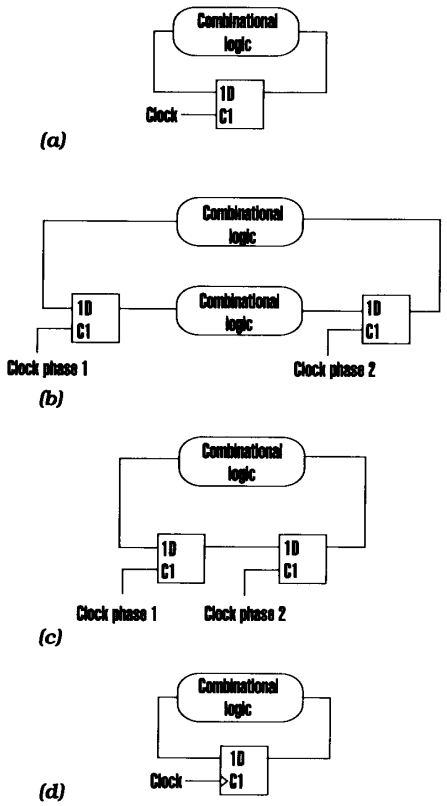

Figure 2. *General finite-state machine structures: one-phase latch machine* ***(a)****, two-phase latch machine with single-latch* ***(b)*** *and double latch* ***(c)****, and flip-flop machine* ***(d)****.*

THE CLOCK CYCLE

System designers characterize a computer's functionality in terms of its clock cycle, also called its machine cycle. The average number of clock cycles required per machine instruction is a measure of computer performance. Table 1 gives clock rates for some well-known systems. The designer focuses on the clock cycle because it determines the standard work interval for internal machine functions. The system state is the set of values in system memory elements at the end of a clock cycle.

A clock cycle has the following properties:

1. It consists of a sequence of one or more clock pulses.
2. The sequence of clocks generated in each cycle is identical to every other cycle.
3. No partial clock sequences can occur: clocks can only stop and start at cycle boundaries.
4. Each bistable element can be updated at most once per cycle.

These properties ensure that the transition to the next state of the system is predictable and correct. This deterministic system

In a conventional computer system, one source generates the system clock signal. Multiple processors operating synchronously may also share one signal.

behavior will hold whether clock cycles occur at the system operating rate or one at a time. We can reproduce system behavior at the operating rate by issuing single clock cycles or bursts of clock cycles, which makes system debugging much simpler.

TIMING ANALYSIS

Programs for timing analysis are used routinely to verify system timing. They can identify long or short paths, and the designer can interact with them to get estimates of signal-path delays in parts of the system. Designers can also run them after layout to get more accurate results. The delay models used for system elements are validated by circuit simulation.

Single-phase systems and multiphase overlapping systems require more extensive timing analysis than multiphase nonoverlapping and edge-triggered systems. The timing constraints of single-phase and multiphase overlapping systems are two-sided, bounded by both short paths and long paths. Figure 3 illustrates these constraints in a simplified example, where setup time and hold time are set to 0. The advantage of these systems is that they operate more quickly than their nonoverlapping counterparts.

CLOCK SIGNALS

In a conventional computer system, one source generates the system clock signal. Multiple processors operating synchronously may also share one signal. We can manipulate this clock signal in many ways before it reaches its destinations. We can divide it, delay it, shape it, buffer it, and gate it. Clocked bistable elements, either latches or flip-flops, use the signal that results from such manipulations.

Table 1. *System clock rates.*

System	Intro Date	Technology	Class	Nominal Clock Period (ns)	Nominal Clock Frequency (MHz)
Cray-X-MP	1982	MSI ECL	Vector processor	9.5	105.3
Cray-1S,-1M	1980	MSI ECL	Vector processor	12.5	80.0
CDC Cyber 180/990	1985	ECL	Mainframe	16.0	62.5
IBM 3090	1986	ECL	Mainframe	18.5	54.1
Amdahl 58	1982	LSI ECL	Mainframe	23.0	43.5
IBM 308X	1981	LSI TTL	Mainframe	24.5,26.0	40.8,38.5
Univac 1100/90	1984	LSI ECL	Mainframe	30.0	33.3
MIPS-X	1987	VLSI CMOS	Microprocessor	50.0	20.0
HP-900	1982	VLSI NMOS	Micro-mainframe	55.6	18.0
Motorola 68020	1985	VLSI CMOS	Microprocessor	60.0	16.7
Bellmac-32A	1982	VLSI CMOS	Microprocessor	125.0	8.0

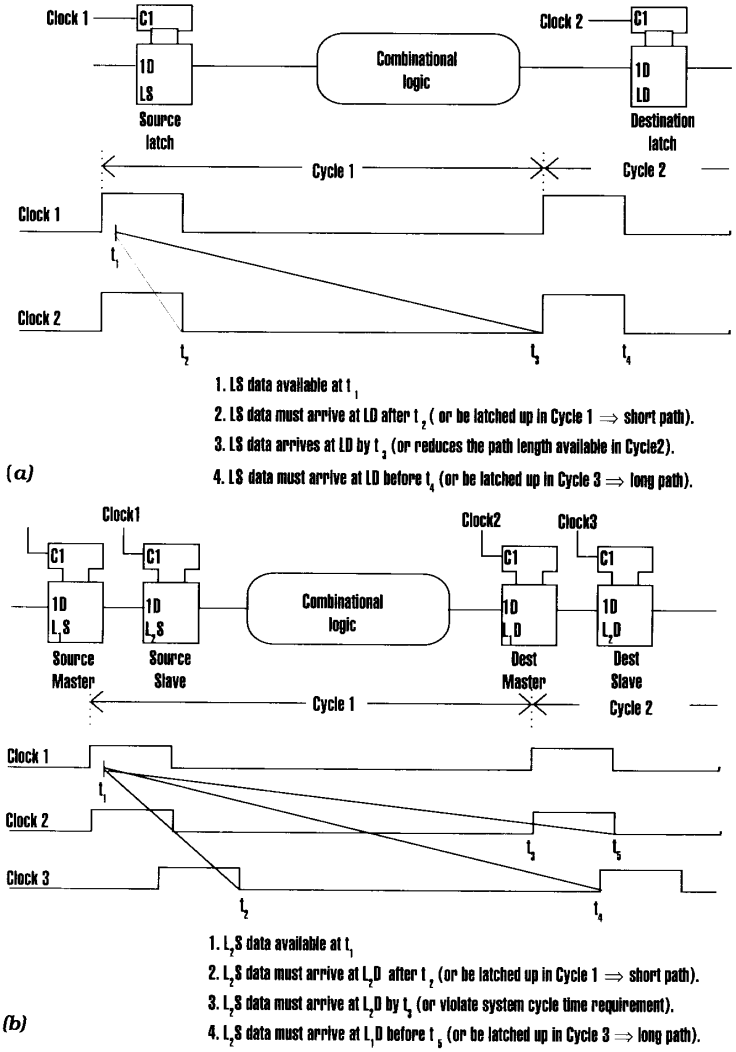

Figure 3. *Path requirements in a single-phase machine* ***(a)*** *and in a two-phase overlapping latch machine with a double latch* ***(b)***.

For all systems, we must correctly place the leading- or trailing-edge positions of the distributed clock pulses to ensure that bistable elements switch at the correct times.

SIGNAL CHARACTERISTICS

Clocked sequential logic responds to several characteristics of the clock signal: the clock period, the pulsewidth, and the leading-edge or trailing-edge position of the clock pulse. The *clock period* is the interval before the signal pattern repeats. The ideal clock signal for a bistable element is a sequence of regularly repeating pulses. Ideal pulses are rectangular with sufficient duration and amplitude to ensure the reliable operation of the bistable element. The duration of the pulse, or pulsewidth (W), can be any fraction of the clock period, but is usually less than or equal to half of it. An accurate model of a real clock pulse includes actual voltage levels and the shapes of the pulse edges.

Pulsewidth-manipulation elements have three functions: chop, shrink, and stretch.

For all systems, we must correctly place the leading- or trailing-edge positions of the distributed clock pulses to ensure that bistable elements switch at the correct times. Also, distributed clock pulses must be wide enough or they will either be filtered out in transmission or be unable to switch a bistable element because they lack the energy. *Clock-manipulation elements* reposition clock pulses and change their pulsewidths. They consist of delay elements and elements that manipulate the pulsewidth. *Delay elements* either delay a pulse, or, in a timing chain, produce a sequence of delayed pulses in response to a single pulse input. *Pulsewidth-manipulation elements* require both delay elements and logic gates.

Delay elements are available as both analog and digital circuits and are chosen according to the accuracy, flexibility, and range of signal delay required. Analog delay elements vary from simple printed or discrete wire interconnections to delay lines. Delay lines, packaged in hybrid chips, consist of lumped LC elements or distributed printed wire, which provides more accurate control. Digital delay elements include logic gates and counters. Logic gates are relatively inaccurate because of their wide delay ranges, while the time resolution of counters depends on their operating frequency.

Some delay elements are programmable, providing a range of delays. To select a particular delay, we can either connect to a particular chip output pin or tap, or control the configuration electronically by a multiplexer. A typical integrated delay line provides delays from 1 to 10 ns in 1-ns increments with a ±0.5-ns tolerance.

Pulsewidth-manipulation elements have three functions: chop, shrink, and stretch. Figure 4b shows the effect of a chopper, shrinker, and stretcher on a positive pulse. The effect of each manipulation element differs for positive and negative clock pulses. Thus, for each pulse polarity, only three of the four elements are useful. The other element has only a delay effect. Table 2 shows the values for the signal characteristics after chopping, shrinking, and stretching. AND gates have delay d_a, OR gates have delay d_o, inverters have delay d_i, delay elements have delay D, and interconnections have no delay. The signal input is a pulse of width W whose leading edge occurs at time t=0. For an element to have an effect during the pulse, the sum of d_i and D must be less than W.

Table 2. *Effect of elements that manipulate the clock pulsewidth.*

Element	Positive Pulse			Negative Pulse		
	Leading Edge	Pulse-width	Function	Leading Edge	Pulse-width	Function
A	d_a	$D+d_i$	Chopper	—	—	—
B	—	—	—	d_o	$D+d_i$	Chopper
C	$D+d_a$	W–D	Shrinker	d_a	W+D	Stretcher
D	d_a	W+D	Stretcher	$D+d_a$	W-D	Shrinker

CLOCK GENERATION

We can derive all clock signals in a synchronous machine from the system clock signal. The system clock is often a rectangular pulse train with a 50% duty cycle, called a *square wave*. The circuit that generates the system clock is at the base of the clock-distribution network. Its input is from either a voltage-controlled oscillator (VCO), a crystal oscillator (XO), or a voltage-controlled crystal oscillator (VCXO). All three sources produce a sinusoidal (single-frequency) output, which is then clamped or divided to generate the rectangular system clock. Excluding the quartz crystal, the oscillator circuit is usually packaged on a single hybrid IC.

A simple oscillator consists of an LC circuit, which we tune by carefully selecting component values that allow the circuit to resonate at the desired frequency. When we need extreme frequency stability over a wide temperature range, we use an XO. An XO consists of a tuned circuit with an embedded quartz crystal in the feedback loop. The crystal stabilizes the resonant frequency of the oscillator circuit.

When we need a larger range of selectable frequencies, we use either a VCO or a VCXO, because the XO has a very limited tunable range. A DC voltage input controls both the VCO and VCXO. The VCO could be an emitter-coupled multivibrator that produces a square wave that we can tune over a 10:1 frequency range up to 20 MHz. It could also be a capacitance-controlled oscillator that produces a sine wave tunable over a 2:1 frequency range up to microwave frequencies. If we modify the resonant frequency of an XO, we get a tuning accuracy of a few hundred parts per million in the VCXO. Thus, the XO has the most frequency stability but the least tuning flexibility, the VCXO is in the middle on both, and the VCO has the least frequency stability and the most tuning flexibility. Frequency instability in the oscillator can cause clock jitter, requiring us to assign a tolerance to the clock-edge placement in timing analysis.

From the system clock we derive the full set of clocks and clock phases that the system requires. We can generate multiphase clocks from a square-wave input in many different ways. These methods include one shots, clock choppers or shrinkers, shift-register latches, and frequency dividers, depending on the precision and flexibility required. To prevent the overlap of adjacent clock phases in a nonoverlapping clocking scheme, we use output feedback or clock choppers. If there is uneven loading on each clock phase, the relative pulse-edge positions may change, which might cause some of the clock phases to overlap. Another cause of overlap is the asymmetric rising and falling delays of contemporary devices.

Figure 5 shows two simplified circuits that create two-phase clocks. The techniques are applicable to general multiphase clock generation. The first circuit is used in the Univac 1100/90 for four-phase clock generation. It requires a fast-running square-wave clock input and a ring counter. Each stage of the ring counter enables one clock phase, and the single clock chopper

From the system clock we derive the full set of clocks and clock phases that the system requires.

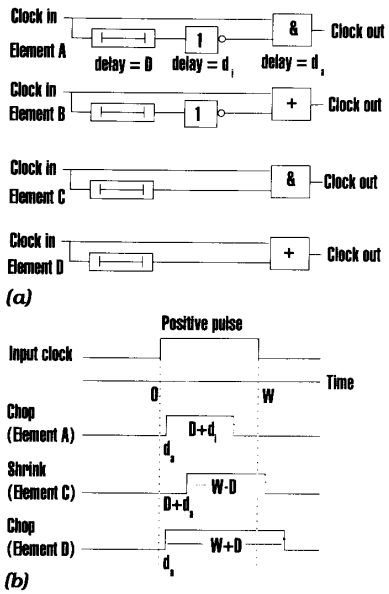

Figure 4. *Elements that manipulate the clock pulsewidth* ***(a)*** *and their effect on a positive pulse* ***(b)****.*

For developing, diagnosing, and producing high-speed systems, we ideally want a wide-bandwidth oscillator source that is highly accurate.

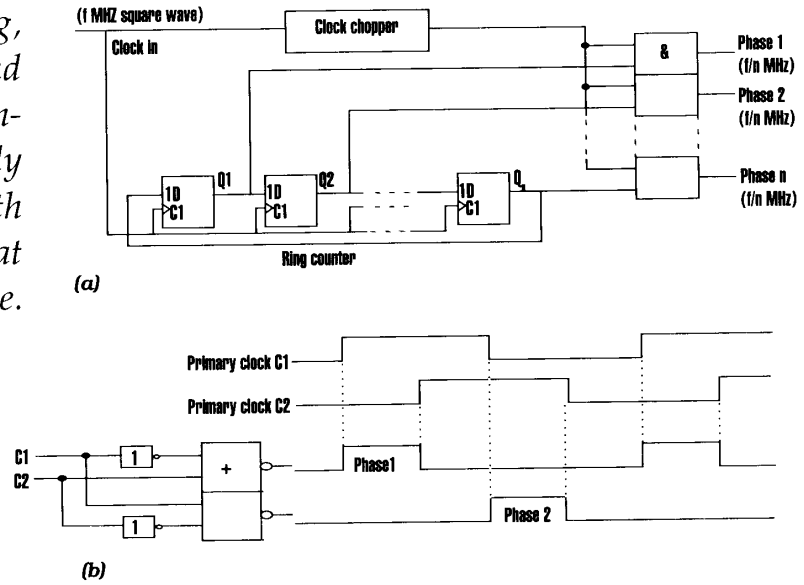

Figure 5. *Creating a two-phase clock: selecting the pulses of a fast-running clock* ***(a)*** *and decoding the primary clocks* ***(b)****.*

determines pulsewidth. The second circuit is used in the Bellmac-32A. It generates two-phase clocks by decoding primary clock signals. We can use a gray-code counter to produce these primary clocks, or we can use *clock shaping*. Clock shaping allows us to generate clock phases from a system clock with fixed gaps between phases (forcing pulsewidths to vary with frequency).

CLOCK SEQUENCES

The three schemes for system clocking we have looked at—single-phase, multiphase, and edge-triggered—determine the basic data flow in latch and flip-flop machines during each clock cycle. Complicating these requirements, though, are special timing considerations. For example, subsystems may require dfferent clock-arrival times so that they can communicate with each other across interfaces with large delays. Also, paths within subsystems may be too long for normal system timing. We can accommodate irregular interfaces and paths without affecting the clock cycle, although system timing becomes more complex. To handle these cases, we generate a sequence of clocks during each clock cycle and do not use normal data-path timing.

There are two timing design styles for handling the clock sequences generated during a clock cycle: multiphase design and multiclock design. Figure 6 illustrates. The dashed vertical lines represent the boundaries of the clock cycle. The solid vertical lines represent active clock edges. Time proceeds left to right across each diagram and only paths originating from the earliest (leftmost) cycle are shown. In a normal multiphase (k-phase) design

(Figure 6a), latches clocked by phase 1 feed latches clocked by phase 2, and so on. Only the latch clocked on the last phase feeds the phase-1 latch of the succeeding cycle. All data movement proceeds phase i to phase $i+1$ modulo k.

In contrast, the multiclock design (Figure 6c) ensures that bistable elements clocked at any time T_i during one cycle feed only bistable elements clocked in the succeeding cycle. For instance, the three cycle $n-1$ clocks are early, normal and late, which correspond to the times T_0, T_1 and T_2. Each can feed any of the T_0, T_1 or T_2 bistable elements in cycle n.

In the Amdahl 580, early clocks prevent long paths between the remote channel frame and I/O processor. If we clock the source latch earlier or destination latch later than normal on a signal path, the signal has a longer interval to propagate. Of course, other signal paths between latches using normal clocks as sources and early clocks as destinations will have a shorter than normal time to propagate. Similarly, paths with latches using late clocks as sources and normal clocks as destinations will also be shorter.

Multiphase design and multiclock design can be mixed, as shown in Figure 6d. The two-phase, double-latch configuration has master latches, which feed their associated slaves in the same cycle. Each master latch is clocked at one of three timings: T_0, T_1 or T_2. The slave latch of each pair communicates with any of the master latches in the next cycle. The IBM 3033, 308X, and 3090 mainframes use similar techniques.

Figures 6b, 6e, and 6f show examples of more complex paths. Figure 6b shows the possibility of paths that skip adjacent phases in a three-phase system. The Univac 1100/90 is an example of a design with nonadjacent phase paths. Note that any phase-i-to-phase-i path in the succeeding cycle would require identical analysis to a single-phase system. Figures 6e and 6f show fractional cycle and multicycle paths. Such paths are typical of a performance-oriented design that uses two-phase latch machines.

Systems can also generate clocks that operate at several distinct cycle times, usually integer multiples of a base cycle time. We can use clocks with lower rates for parts of the system that do not need faster clocks. All clocking between subsystems must be synchronous, or else we must use techniques to reduce metastable behavior at subsystem interfaces.

THE SYSTEM CLOCK SOURCE

For developing, diagnosing, and producing high-speed systems, we ideally want a wide-bandwidth, highly accurate oscillator source. Most systems have both a crystal-oscillator source input for production systems and a tunable source input for prototype development and AC diagnosis. During development of a multiphase system, we may need to vary the pulsewidth of any clock phase as well as to vary the relative pulse positions.

To detect marginal path-delay problems, the Amdahl 580 selects any one of three crystal oscillators as the clock source in production machines, lengthening or shortening its clock cycle.

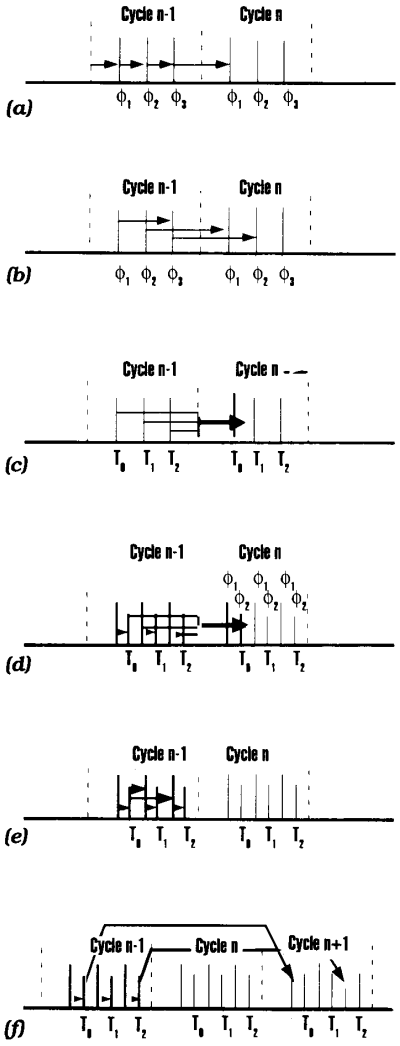

Figure 6. *Placing clock pulses; three-phase, adjacent paths* ***(a)****; three-phase, nonadjacent paths* ***(b)****; multiclock (three clocks)* ***(c)****; multiclock, two-phase* ***(d)****; multiclock, two-phase with fractional cycle paths* ***(e)****; and multiclock, two-phase with multicycle paths* ***(f)****.*

The goal of clock distribution is to organize clocks so that the delays from the source of each clock or clock phase to its bistable elements are identical.

Operating modes are called normal, fast margin, and slow margin. These correspond to nominal clock frequency, 5% faster than nominal, and 5% slower than nominal. An external oscillator input is also available, bypassing the internal oscillators during diagnosis and development.

To detect marginal timing problems in the IBM 3090, a two-phase double-latch machine, designers made it possible to lengthen the delay between the leading edge of the slave clock and the trailing edge of the master latch clock for a selected system region (see Figure 3b). In addition, lengthening the clock cycle allows us to verify the slave-latch-to-slave-latch path delay.

We can choose between distributed or centralized clock sources to control multiprocessors synchronously. In distributed control, we let each processor or processor group in the complex use its own local oscillator, with some form of enforced synchronization between oscillators, like a phase-locked loop. Alternatively, in centralized control, we designate one oscillator as the master oscillator and have each system select this master through a local/remote switch. The second method is simpler and is common in mainframe multiprocessor models such as the Amdahl 580, IBM 3033, and IBM 370/168. Although the IBM multiprocessors use a master oscillator, other standby oscillators are phase-locked to the master oscillator and can be selected if it fails.

CLOCK DISTRIBUTION

The goal of clock distribution is to organize clocks so that the delays from the source of each clock or clock phase to its bistable elements (its destinations) are identical. In reality, however, no matter how each clock path is constructed, any two clock paths in the same machine or any two corresponding paths in different machines will always have a delay difference. Every computer operates in a different temperature, power supply, and radiation environment, and duplicate components will differ in subtle ways between computers. We must build in tolerance to these variations in any system timing design.

The most common approaches to ensure correct and reliable machine timing are worst-case analysis and statistical analysis. In worst-case analysis, we assume that all component parameters lie within some range, and the cumulative worst-case effect is still within the timing tolerance of the machine. In statistical analysis, the intent is that most machines have tolerable timing characteristics, and so we can rely on the cumulative statistical variations of component parameters to remain within the timing tolerance.

CLOCK SKEW

We specify system timing such that every system memory element has an expected arrival time for the active edge of its clock signal. Clock-edge inaccuracy is the difference between the actual and expected arrival time of this clock edge. For every pair of system memory elements that communicate, we define *path*

clock skew as the sum of the clock-edge inaccuracies of the pair's source and destination. *System clock skew* is the largest path clock skew in the system. It is the value of the worst-case timing inaccuracy among all paths. We can break it into interboard skew, on-board interchip skew, and so on to the smallest timed component.

The challenge to designers of clock-distribution networks is how to control system clock skew so that it becomes an acceptably small fraction of the system clock period. As a rule, most systems cannot tolerate a clock skew of more than 10% of the system clock period. If system clock skew goes beyond the design limit, system behavior can be affected. Setup and hold times are missed, which results in long and short paths. No scheme is immune from these problems—even flip-flop machines can malfunction when clock skew is present.

Clock-powering trees, such as the one in Figure 7, are a source of clock skew. These trees are used to produce multiple copies of the clock signal for distribution. Each gate of the tree has some uncertainty associated with its delay, which is the difference between its best-case and worst-case delays. This difference is called *gate skew*. Using a worst-case timing analysis, the clock skew caused by a powering tree equals the arithmetic sum of the gate (and interconnection) skews on the path from the tree root to an output. In other words, clock skew has a cumulative effect by tree level. We can minimize this clock skew by placing all gates at a given tree level, or even the entire tree, on the same chip. In addition, we can realize elements at each tree level by using electrically matched devices and careful wiring.

Critical to efficient distribution is the clock-network layout—the physical placement of the network.

DISTRIBUTION TECHNIQUES

We must efficiently distribute the rectangular clock pulses produced through the interaction of the oscillator and clock-generation circuitry. Critical to efficient distribution is the clock-network layout—the physical placement of the network. It must conform to design rules that ensure the integrity of the clock signal by minimizing electrical coupling, switching currents, and impedance discontinuities. Other rules must prevent excessive clock skew by equalizing path delays and maintaining the quality of the signal edge. Symmetry and balanced loading at many levels of packaging, such as on the chip or on the board, are characteristic of effective clock-network layouts. To achieve these qualities, we can prearrange positions of the clock pins and make clock paths as short as possible.

It is sometimes difficult to coordinate the relative lengths of two paths that originate from a common source. To match any two paths or path segments in the clock system, we may need identical lengths of cable, wire, and interconnections; balanced loading; and equal numbers of buffer gates. A technique called *time-domain reflectometry* helps in this process by accurately measuring the line delays of long cables. In this process, line sig-

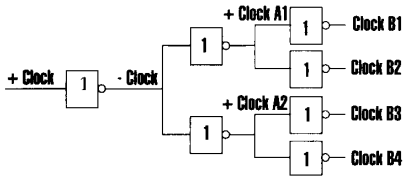

Figure 7. *Clock-powering tree.*

In practice, large systems distribute a small number of clock signals to each board or module.

nals are generated, and the signal reflections from line terminations are detected in real time. Once we measure the delays, we can equalize them by adjusting the lengths of the cables.

Duplicating the composition of two paths is not the only method of ensuring that two paths have equal delay. Another technique for matching different paths is called *padding*. In padding, we add delay elements to one or both paths. The Cray-1 uses extra interconnections and spare IC packages as padding, for example.

System designs often use a mixture of strategies. Designers might use duplication in subsections of the clock-distribution network with intermediate padding. Component screening also ensures that the performance of each system component is acceptable. Despite these techniques, some system clock skew is inevitable. However, as long as the timing analysis includes the effect of clock skew and determines that the system will function correctly, no other precautions are needed. If we detect that some system paths are failing because of excessive clock skew, then the clocks have to be tuned, or some part of the network has to be redesigned.

High-speed systems use a hierarchical structure to distribute clocks efficiently. A model for such a structure consists of *logic islands*. A logic island is a partition in the system, like a printed circuit board. Each island has a single point for clock entry, and all islands have the same line delay from their clock source, outside the island, to their clock entry point. (Tunable delay lines may precede the clock entry points.) We apply the same technique recursively to the islands themselves and to the subislands, such as chips, until we reach the individual clocked elements. Figure 8 is a diagram of the resulting structure, which resembles a star with the clock source in the center and the islands on the periphery. This model is remarkably similar to the physical organization of the Cray-1. In systems that need multiphase clocks, clock phases are generated at some convenient level in the hierarchy, and clock entry points at subsequent levels have one input for each clock phase.

In practice, large systems distribute a small number of clock signals to each board or module. Either the leading- or trailing-edge position of these clocks is very tightly controlled. When these signals reach board or module distribution, phase-generation circuits and clock choppers produce the final, well-controlled pulses. With this strategy, we can simplify clock distribution and reduce clock skew because only one clock edge has to follow a predetermined relationship to a reference clock. Using a single edge also minimizes clock skew by exploiting *common-mode action*, in which changes in power and temperature have similar effects across clock circuitry.

Clock choppers should be close to the bistable elements that are the signal's final destination. Otherwise, asymmetric rising and falling delays in the buffer gates downstream of the clock chopper may shrink or stretch the pulse excessively. For instance, in the IBM 3090's two-phase double-latch design, the leading edge of the slave clock and trailing edge of the master clock are formed from a common input clock edge. These edges control the

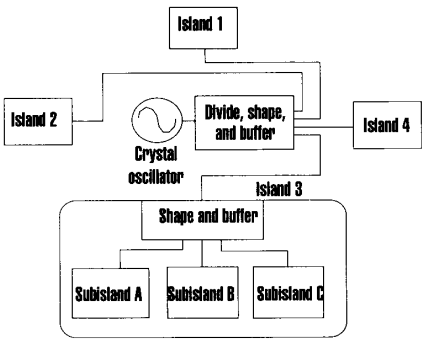

Figure 8. *Logic islands.*

critical system path, so they must have minimal skew. Exploiting common-mode action in this way is also known as edge-tracking.

There are a number of simple guidelines for on-chip clock distribution to minimize asymmetries in clock-path delays and keep clock edges sharp. We balance and limit on-chip clock loading, using clock-powering trees, special buffers, symmetric layouts, and careful buffer placement. In very high speed ICs, we can select H-trees, which are symmetric, controlled-impedance clock-distribution trees composed entirely of metal. We can also reduce process-dependent clock skew in CMOS chips by adjusting clock-buffer FET parameters appropriately.

DESIGN DECISIONS

One of the first decisions in the design of the clock-distribution network is where to put square-wave production—the function that will produce the system clock—and where to put the function that will generate additional clocks. If a system requires many different clocks, we should distribute only a small number of clocks globally, and then generate the necessary clocks for each system section locally by manipulating (including decoding) these primary clocks.

Most mainframes use gate arrays because designers need to trade off chip turnaround time with the large number of different chips required.

In systems constructed from gate arrays or other semicustom chips, we cannot manipulate device characteristics individually. Most mainframes use gate arrays because the designers need to trade off chip turnaround time with the large number of different chips required. Both MOS and bipolar gate arrays have limited fanout of clock driver circuits. Whole chips or portions of chips are dedicated to controlling clock signals and buffering clock signals through powering trees (see Figure 7). The designers' intent is to produce as many copies of the clock signal as necessary to satisfy the loading requirements of the system. The more system bistable elements and associated chips and the smaller the amount of fanout allowed for each clock copy, the larger the number of required tree levels.

Figure 9 shows a simplified example of the clock-distribution network in the Amdahl 580. In this mainframe, which is based on bipolar gate arrays, the circuitry that produces the system clock is on the same card as the oscillator. Special chips on the console board then receive the system clock. These chips do clock gating and powering to provide the primary clock input for all boards. On the boards, tunable clock-distribution chips receive the primary clock and do clock chopping; derive the early, normal, and late clocks; and power these clocks sufficiently to satisfy loading requirements. The Amdahl 580 uses two ECL LSI clock-distribution chips to drive about 118 ECL LSI logic chips and static RAMs on each of its boards.

Custom systems and those constructed from catalog parts also use separate clock buffer chips when clock signals must drive large capacitive loads. Dedicated chips produce the clock copies that the system needs. Buffers on these chips must supply large

To reduce clock loading and allow for local clock gating, we can provide groups of bistable elements with their own small, local buffers.

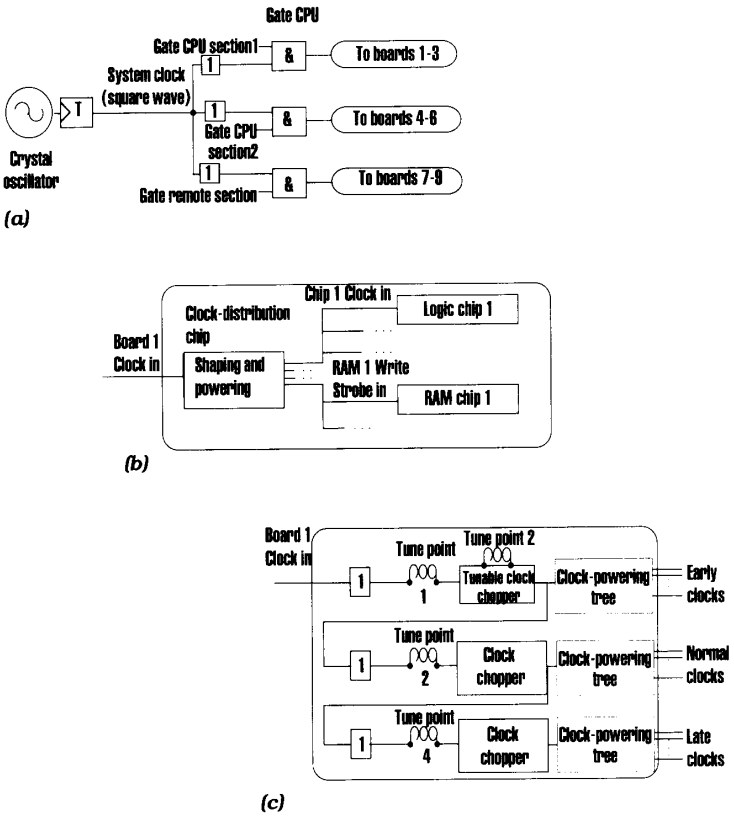

Figure 9. *Clock-distribution network with system clock gating: system-wide clock distribution* ***(a)****, board-clock distribution* ***(b)****, and clock-distribution chip* ***(c)****.*

currents, and accordingly, each buffer occupies a large area of each buffer chip. Buffer chips in the HP-9000 also generate clock phases.

In MOS custom chips or chip sets like the MC68000, we can produce clock phases on the chip and then buffer them immediately to provide adequate fanout. Most MOS single-chip systems use a single or cascaded clock buffer as a source for each distributed clock phase. Each buffer drives a very wide metal line (large load). The line composition and dimensions minimize the clock signal's power loss and voltage drop, while maximizing the speed of signal propagation. The wide metal line branches into groups of narrower metal lines which themselves may branch off, and so on. Nonmetal segments with a higher resistance, like polysilicon, should be avoided in clock-distribution lines. The distortion of the clock signal increases when it is propagated across such impedance mismatches. In these regions, clock phases may overlap and violate system timing constraints.

In single-level metal processes, power distribution has priority on the single metal layer available. Clock lines forced to cross the power lines cannot be run in metal. In the CMOS Bellmac-32A, for example, designers ran clock lines that crossed the power bus in a low-resistance silicide, and they kept the number of crossovers to a minimum. To reach any clock load, they perform the same number of crossovers, which equalizes all the resulting path delays. In addition, they provided buffers at the crossovers to minimize these delays. In spite of the Bellmac 32-A's success, however, multilevel metal processes are a prerequisite to very high speed systems with low clock skew.

Some designers manage to avoid clock tuning by carefully designing and routing the clock network.

CLOCK GATING

Selectively deactivating the clock signal is called *clock gating.* Designers can use one of two types of clock gating, depending on the application. *Local clock gating* is a convenient way to implement many sequential circuits by locally deactivating the clock to a set of bistable elements, such as a register. To reduce clock loading and allow for local clock gating, we can provide groups of bistable elements with their own small, local buffers. The drive capabilities and numbers of on-chip clock buffers can be matched to the loading on their outputs. In *system clock gating,* the clock to an entire subsystem is deactivated. Figure 9a illustrates. In the Amdahl 580 and HP-3000, system-clock gating is done at the board level, while the Amdahl 470 does clock gating on the oscillator card. Gating signals must be valid through the entire active clock interval to prevent glitches on the gated clock line that could be sensed as valid clock pulses.

System clock gating is a tool for analyzing errors and recovering from them. It allows us to test the machine in a deterministic fashion by ensuring that a predetermined minimum number of clock cycles occurs after some machine stop condition. This type of testing makes it easier to isolate faults. For example, in the IBM 308X, the operator console deactivates the channel subsystem and logs out its contents through scan-out. It then scans in to reset the failed section. Amdahl 580 mainframes have separate console clocks, instruction and execution unit clocks, and I/O processor clocks, all of which are separately gated versions of a common, ungated system clock (also called a free-running clock).

CLOCK TUNING

High-speed computer systems with multiple boards and many chips on each board often require clock tuning after assembly. Clock tuning is calibrating the signals of the clock-distribution network. Some designers manage to avoid clock tuning by carefully designing and routing the clock network. Clock tuning during assembly and in the field is an expensive process, both in time and in the cost of the technical expertise. Because of this, designers must minimize the number of clock-tuning operations.

Tuning proceeds down the clock-distribution tree from clock source towards the clock destination.

We can determine how much clock tuning is needed by comparing clock signals probed at specified observation points. Modification of delays in the clock-distribution path then compensates for any significant inaccuracy in the clock-edge position or pulsewidth. Tuning can be manual, automatic, adaptively automatic, or a combination.

REFERENCE CLOCKS

To specify the placement of clock edges in a system, we designate one or more of the system's clocks as reference clocks. We use transitions of these timing references, or reference-clock edges, for comparison with other clock edges. We can specify the arrival time of a clock signal at any particular point in the clock-distribution network relative to these reference clocks.

TUNE POINTS

Tuning proceeds down the clock-distribution tree from clock source towards the clock destination. By tuning in this direction, we have the fewest number of tuning operations to calibrate the system because there is no backtracking. We can tune large components, such as printed circuit boards, separately to reduce the tuning requirements of the fully assembled system.

Tune points are the observation points in the clock-distribution tree where we can change the delay. Tuning methods to reposition clock edges include modifying wire lengths, selecting different taps in delay lines, or selecting one delay element from a set by controlling a multiplexer. We can tune clock choppers by adjusting their internal delay elements to control pulsewidth as well.

A tune-point hierarchy is embedded within the clock-distribution network. Figure 10 shows how we can use a tree structure to model this hierarchy. The level of the tune point in the tree is referred to as the depth in the tune-point hierarchy—the deeper the tune point, the farther away it is from the system oscillator. Chip or module primary I/O are usually the deepest accessible tune points.

Designing accessible and effective tune points may be difficult and their tuning resolution and range may be limited. One ap-

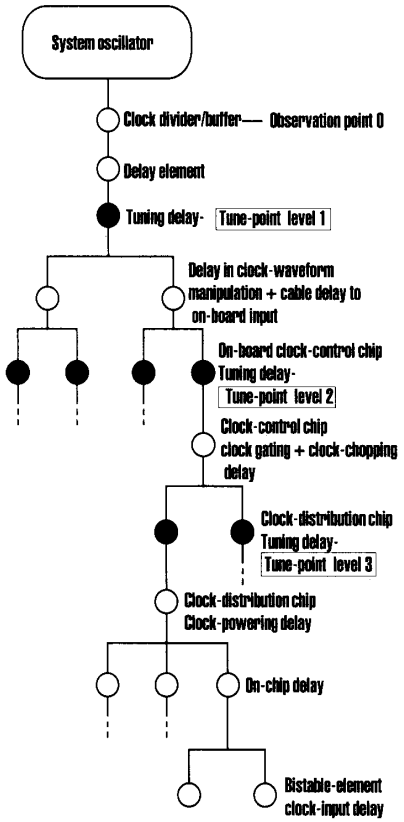

Figure 10. *Tune points.*

Table 3. *Sample clock-specification plan; all clock times are in nanoseconds.*

Parameters	Observation Point Level 0	Tune-Point Level 1	Tune-Point Level 2	Tune-Point Level 3	Tune-Point Level 4
	Observation Point				
Local reference clock edge	0	+3.0	+7.0	+13.0	+20.0
Local tolerance	—	±0.5	±0.5	±0.5	±0.5
Effective clock arrival time	0	3.0±0.5	+7.0±1.0	+13.0±1.5	+20.0±2.0

proach to the problem is to determine the worst-case delay in a clock path and pad other clock paths to match this delay. For instance, in subsections of the clock-distribution network, one path may be significantly longer than all others and not have any tune points. By adding delay elements at tune points, we can pad all other comparable paths, either to have the same or shorter delay, which is fixed by design. A sophisticated physical design system can automatically design in this type of tuning by adding wire and capacitive elements as needed.

System timing is based on a clock-specification plan. The plan details the allowable ranges for clocks at each tune point, relative to a reference clock. We need to place tune points in such a way that the sum of the clock skew at the deepest tune point plus the additional skew for the signal to reach bistable elements beyond that tune point does not exceed the acceptable limit on system clock skew. Table 3 shows a sample clock-specification plan for Figure 10. The uncertainty is ±0.5 ns for each of 13 required tuning operations. In this plan, system clock skew equals 3.0 ns plus the maximum of the clock skews on the paths between the 12 tune points at Level 3 and the bistable elements that their clocks control.

We can provide extra observation points for clock tuning by distributing supplementary clock signals for use as precise reference clocks.

TUNING SCHEMES

Tuning schemes vary in sophistication. Manual tuning by a trained technician using an oscilloscope is common. The technician calibrates the tune points sequentially, starting from the one closest to the system clock source. We can provide extra observation points for clock tuning by distributing supplementary clock signals for use as precise reference clocks. No powering trees or other skew-increasing elements are allowed in the signal paths of these clocks, so they have little or no need for tuning themselves. We can thus use them safely and tune all other clock signals relative to the these references. This strategy increases tuning accuracy and decreases the number of tune points in the system.

Knowing the clock signal internal to the chip is often helpful in tuning. For MOS systems with multiple chips, on-chip clock buffering creates uncertain delays, so we must observe a representative internal clock signal. We can use this internal reference clock, which is output at a chip pin, as both a functional clock signal (for small loads), and as a reference to be compared with the corresponding references of other chips. We can then use the relative edge positions of the internal reference clocks to guide tuning.

In the clock-tuning schemes of the ETA 10 supercomputer and IBM 4341, designers provided two separate tuning resolutions, or tuning levels: rough tuning and fine tuning. Fine tuning is required for minute adjustments deep in the clock tree, while rough tuning provides the coarse adjustment earlier in the distribution and (in the ETA machine) before the system is immersed in coolant.

Untuned systems —designed with attention to component variations and to equalizing wire lengths and clock loading —eventually proved less expensive and entirely adequate.

A typical automatic tuning technique, devised for the discontinued STC CMOS mainframe, uses clock-distribution chips with many degrees of time-shifted clocks available through a large crossbar switch. In this case, a logic chip has four internal reference clocks. Each internal reference clock is a representative internal clock produced after chopping and powering one of four chip clock inputs. These representative clocks are compared with a precise reference clock. The correct skew-minimizing clock for each chip clock's primary input is then selected automatically from the crossbar switch. Registers on the clock-distribution chips are loaded to properly configure each crosspoint of the switch.

Automatic tuning often consists of closing a special feedback connection in a clock network that has an odd number of inversions in the signal path. When this connection is closed, we get oscillations with a period proportional to the total delay of the path. If necessary, we can adjust one or more tune points automatically through control signals determined by diagnostic code. Feedback tuning techniques seem attractive, and numerous patents exist for them, but their sensitivity to the clock duty cycle and the signal transition time make them complex and usually impractical.

Thus, more sophisticated tuning schemes are not necessarily better ones. Proposals for system clocking of the IBM 308X and 3090 mainframes required complex automatic tuning techniques. Eventually, untuned systems—designed with careful attention to component variations and to equalizing wire lengths and clock loading—proved less expensive and entirely adequate.

The clock system is an integral part of synchronous computers, yet it is not a widely studied aspect of their design. Early attention to system timing issues can provide benefits in system performance as well as product development time. The goal of the clock system designer is to control system clock skew at the system operating frequency as well as to minimize electrical hazards that may add undesirable components to the clock signals. Familiarity with clock-generation and clock-distribution techniques suitable for high-speed systems is essential as cycle times decrease.

ACKNOWLEDGMENTS

This tutorial was supported in part by the National Sciences and Engineering Research Council of the Government of Canada under its postgraduate scholarship program, in part by the National Science

Foundation under grant DCR-8200129, and in part by IBM Corp. The work was performed at Stanford University's Center for Reliable Computing and IBM.

I thank Edward McCluskey and Mark Horowitz of Stanford University, as well as Glen Langdon, Jr., and John DeFazio of IBM Corp., and Ron Kreuzenstein of Amdahl Corp. for their many helpful comments and suggestions.

ADDITIONAL READING

Bakoglu, H.B., J.T. Walker, and J.D. Meindl, "A Symmetric Clock Distribution Tree and Optimized High-Speed Interconnections for Reduced Clock Skew in ULSI and WSI Circuits," *Proc. IEEE Int'l Conf. on Computer Design*, 1986.

Domenik, S., "On-Chip Clock Buffers," *Lambda*, 1st qtr., 1981.

Friedman, E., and S. Powell, "Design and Analysis of a Hierarchical Clock Distribution System for Synchronous Standard Cell/Macrocell VLSI, *IEEE J. Solid-State Circuits*, Vol. SC-21, No. 2, 1986.

Glasser, L., and D. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, chapt. 6, Addison-Wesley, Reading, Mass., 1985.

Hitchcock, R., Sr., "Timing Verification and the Timing Analysis Program," *Proc. Design Automation Conf.*, 1982.

IBM 3033 Processor Complex TO/DM, IBM Corp., Mechanicsburg, Pa., 1981.

Kogge, P., "Hardware Design and Stage Cascading," *The Architecture of Pipelined Computers*, chapt. 2, McGraw-Hill, New York, 1981.

Langdon, G., Jr., *Computer Design*, appendices C and D, Computeach Press, 1982.

Lob, C., and A. Elkins, "HP-9000: 18-Mhz Clock Distribution System," *HP J.*, Aug. 1983.

Maini, J., J. McDonald, and L. Spangler, "A Clock Distribution Circuit with a 100-ps Skew Window," *Proc. Bipolar Circuits and Technology Meeting*, 1987.

McCluskey, E.J., *Logic Design Principles: With Emphasis on Testable Semicustom Circuits*, chapts. 7-8, Prentice-Hall, Englewood Cliffs, N.J., 1986.

Seitz, C., "System Timing," *Introduction to VLSI Systems*, chapt. 7, C. Mead and L. Conway, eds., Addison-Wesley, Reading, Mass., 1980.

Shoji, M., "Electrical Design of the BELLMAC-32A Microprocessor," *Proc. Circuits and Computers Conf.*, 1982.

Shoji, M., "Elimination of Process-Dependent Clock Skew in CMOS VLSI," *IEEE J. Solid-State Circuits*, Vol. SC-21, No. 5, 1986.

Unger, S., and C.J. Tan, "Clocking Schemes for High-Speed Digital Systems," *IEEE Trans. Computers*, Vol. C-35, No. 10, 1986.

Wagner, K.D., *A Survey of Clock Distribution Techniques in High-Speed Computer Systems*, tech. rpt. 86-309, CSL Stanford Electronics Lab., CRC 86-20, Stanford Univ., Stanford, Calif., 1986.

Kenneth D. Wagner is an advisory engineer with the EDS VLSI Design Rules Control Department of IBM, where his research interests are the timing and testing of high-speed systems, including clock generation and distribution, random testing, and design for testability. Previously, he worked for Stanford University's Center for Reliable Computing and for Amdahl Corp. as a systems design engineer.

Wagner received a BEng (Honors) in electrical engineering from McGill University in Montreal and an MSEE and a PhD from Stanford. He also received a four-year NSERC postgraduate scholarship from the Government of Canada. He is a member of the IEEE Computer Society and Sigma Xi.

Wagner's address is IBM Corp., EDS VLSI Design Rules Control Dept., B56/901-3, PO 950, Poughkeepsie, NY 12602.