A typical high-fan-out clock tree has approximately 0.5 to 1.0 ns of skew resulting from multiple buffers and PLLs, each contributing several hundred picoseconds of skew and jitter. The question becomes how much skew a system can tolerate. Skew tolerance is a function of the clock frequency, the trace length separating devices, and IC timing specifications. If a clock tree's skew is relatively independent of frequency, it follows that skew becomes a greater percentage of the clock period, and therefore a greater concern, as the frequency increases.

There is no magic frequency at which clock tree skew and jitter become a dominant factor. Input hold time requirements are often the dominant restriction on clock skew. Evaluating hold time compliance requires knowledge of the source flop's output hold time, t_{OH}, which specifies how long the output remains unchanged after the active clock edge, usually the rising edge. Manufacturers do not always specify t_{OH}, requiring conservative estimates for proper analysis. A conservative estimate is generally 0 ns, or saying that the output can change immediately after the active clock edge.

Consider a hypothetical system using ICs with output specifications of output hold time (t_{OH}) = 2 ns and t_{CO} = 5 ns, and input specifications of t_{SU} = 2 ns, and t_H = 1 ns. The system bus wires have propagation delays ranging from 1 to 2 ns. Skew and jitter are handled as a combined entity in this example.

Below a certain frequency, the skew budget is limited by the hold time requirement,

$$t_{SKEW} < t_{OH} + t_{PROP(MIN)} - t_H$$

$$t_{SKEW} < 2 \text{ ns}$$

The hold time expression is derived knowing that the destination flop's input is delayed from the source flop's output by at least the minimum propagation delay of the interconnecting wires. Wiring delay plus the source flop's t_{OH} provides information on how long the destination flop's input remains static after the active clock edge. This static time minus t_H provides the margin with which t_H is met. Clock skew must be less than this margin for valid timing.

Clock skew as restricted by setup time is calculated by determining the worst-case delay from the active clock edge until the destination flop's input changes. This worst-case delay is the sum of the source flop's t_{CO} and the maximum propagation delay of the wiring. Next, t_{SU} must be added in to satisfy the destination flop. This entire delay restriction is subtracted from the actual clock period to yield the timing margin. Clock skew must be less than this margin. At an arbitrarily chosen frequency of 80 MHz (T = 12.5 ns), the skew budget as a function of setup time is greater than that of hold time,

$$t_{SKEW} < T - (t_{CO} + t_{PROP(MAX)} + t_{SU})$$

$$t_{SKEW} < 3.5 \text{ ns}$$

Readily available clock distribution ICs can be used in this system at 80 MHz and below without concern. At 100 MHz (T = 10 ns), the skew budget decreases to 1 ns, which is achievable, although with little added margin. This method of timing clearly runs out of gas at higher frequencies.

Traditional synchronous clocking does not work at high frequencies, because wire delays and clock tree skew do not scale with advancing semiconductor technology. *Source-synchronous* bus clocking architectures enable operation well in excess of 100 MHz by removing absolute wire delays and clock tree skew as variables in timing analysis. A source-synchronous interface distributes clock and data together as shown in Fig. 16.16 rather than separately as in a conventional synchronous clock tree design. The data source, or transmitter, emits a clock that is in phase with the data as if it were just another signal on the synchronous interface. In doing so, the clock and data have closely matched delays through the IC's output circuits and package. The signals are routed on the circuit board with length matching so that each wire that is part of the interface has approximately

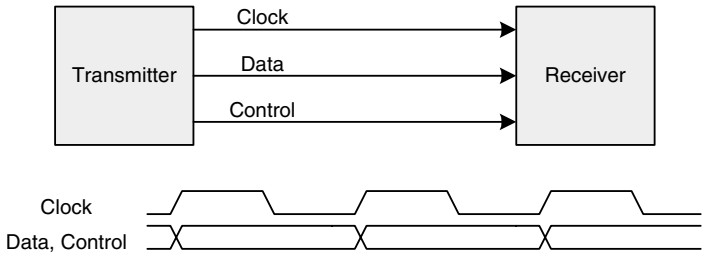

FIGURE 16.16 Source-synchronous bus.

uniform propagation delay. By the time the signals reach the receiving IC, clock and data are phase aligned within a certain skew error that is a function only of the transmitter's output-to-output skew and the delay mismatch of the PCB wiring. This skew is far less than the skew and wiring delay penalties of a conventional synchronous interface and may be on the order of 100 ps. Nearly the entire clock period, less allowances for finite skew and switching time, can be used to meet the input flops' setup and hold specifications, which is why high-performance memory (e.g., DDR SDRAM) and logic interfaces can run at hundreds of megahertz.

The clock and data timing relationship can be arbitrary as long as the skew is tightly controlled, because the receiver's input circuitry can implement appropriate delays based on the transition of data relative to clock to meet the flop's true setup and hold requirements. Two common source-synchronous timing relationships are shown in Fig. 16.17, where the clock is offset into the middle of the data valid window. In the case of single-data rate bus (SDR), this is akin to clocking off the falling edge. A double-data rate bus (DDR) would require that the clock be shifted by 90° relative to data such that the rising and falling edges appear in the middle of the valid windows so that data can be registered on both edges.

A disadvantage of source-synchronous bus architecture is the management of a receive clock domain that is out of phase with the core logic domain. A conventional synchronous design seeks to maintain a uniform clock phase across an entire design, whereas source-synchronous design explicitly gives up on this and proliferates disparate clock domains that have arbitrary phase relationships with one another. This trade-off in clock domain complexity is acceptable, because individual gates are cheap on multimillion-transistor logic ICs. A logic IC typically requires a FIFO and associated control logic to cross between the receive clock and internal clock domains.

Source-synchronous interfaces are often point-to-point, although this is not strictly necessary, because of the high speeds at which they usually run. It is electrically difficult to fabricate a multidrop

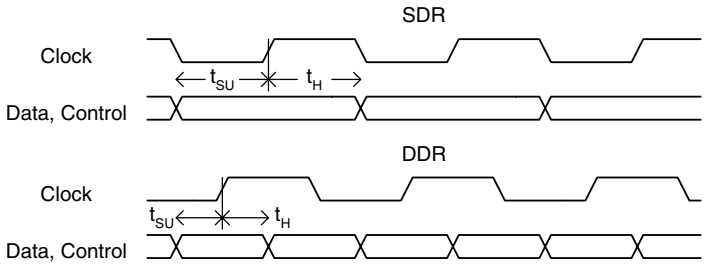

FIGURE 16.17 Typical clock/data timing.