3.5 TIMING OF SEQUENTIAL LOGIC

Recall that a flip-flop copies the input D to the output Q on the rising edge of the clock. This process is called *sampling* D on the clock edge. If D is *stable* at either 0 or 1 when the clock rises, this behavior is clearly defined. But what happens if D is changing at the same time the clock rises?

This problem is similar to that faced by a camera when snapping a picture. Imagine photographing a frog jumping from a lily pad into the lake. If you take the picture before the jump, you will see a frog on a lily pad. If you take the picture after the jump, you will see ripples in the water. But if you take it just as the frog jumps, you may see a blurred image of the frog stretching from the lily pad into the water. A camera is characterized by its *aperture time*, during which the object must remain still for a sharp image to be captured. Similarly, a sequential element has an aperture time around the clock edge, during which the input must be stable for the flip-flop to produce a well-defined output.

The aperture of a sequential element is defined by a *setup* time and a *hold* time, before and after the clock edge, respectively. Just as the static discipline limited us to using logic levels outside the forbidden zone, the *dynamic discipline* limits us to using signals that change outside the aperture time. By taking advantage of the dynamic discipline, we can think of time in discrete units called clock cycles, just as we think of signal levels as discrete 1's and 0's. A signal may glitch and oscillate wildly for some bounded amount of time. Under the dynamic discipline, we are concerned only about its final value at the end of the clock cycle, after it has settled to a stable value. Hence, we can simply write $A[n]$, the value of signal A at the end of the *nth* clock cycle, where n is an integer, rather than $A(t)$, the value of A at some instant t, where t is any real number.

The clock period has to be long enough for all signals to settle. This sets a limit on the speed of the system. In real systems, the clock does not reach all flip-flops at precisely the same time. This variation in time, called clock skew, further increases the necessary clock period.

Sometimes it is impossible to satisfy the dynamic discipline, especially when interfacing with the real world. For example, consider a circuit with an input coming from a button. A monkey might press the button just as the clock rises. This can result in a phenomenon called metastability, where the flip-flop captures a value partway between 0 and 1 that can take an unlimited amount of time to resolve into a good logic value. The solution to such asynchronous inputs is to use a synchronizer, which has a very small (but nonzero) probability of producing an illegal logic value.

We expand on all of these ideas in the rest of this section.

3.5.1 The Dynamic Discipline

So far, we have focused on the functional specification of sequential circuits. Recall that a synchronous sequential circuit, such as a flip-flop or FSM, also has a timing specification, as illustrated in Figure 3.37. When the clock rises, the output (or outputs) may start to change after the clock-to-Q *contamination delay*, t_{ccq}, and must definitely settle to the final value within the clock-to-Q *propagation delay*, t_{pcq}. These represent the fastest and slowest delays through the circuit, respectively. For the circuit to sample its input correctly, the input (or inputs) must have stabilized at least some *setup time*, t_{setup}, before the rising edge of the clock and must remain stable for at least some *hold time*, t_{hold}, after the rising edge of the clock. The sum of the setup and hold times is called the *aperture time* of the circuit, because it is the total time for which the input must remain stable.

The *dynamic discipline* states that the inputs of a synchronous sequential circuit must be stable during the setup and hold aperture time around the clock edge. By imposing this requirement, we guarantee that the flip-flops sample signals while they are not changing. Because we are concerned only about the final values of the inputs at the time they are sampled, we can treat signals as discrete in time as well as in logic levels.

3.5.2 System Timing

The *clock period* or *cycle time*, T_c, is the time between rising edges of a repetitive clock signal. Its reciprocal, $f_c = 1/T_c$, is the *clock frequency*. All else being the same, increasing the clock frequency increases the work that a digital system can accomplish per unit time. Frequency is measured in units of Hertz (Hz), or cycles per second: 1 megahertz (MHz) = 10^6 Hz, and 1 gigahertz (GHz) = 10^9 Hz.

Figure 3.38(a) illustrates a generic path in a synchronous sequential circuit whose clock period we wish to calculate. On the rising edge of the clock, register R1 produces output (or outputs) $Q1$. These signals enter a block of combinational logic, producing $D2$, the input (or inputs) to register R2. The timing diagram in Figure 3.38(b) shows that each output signal may start to change a contamination delay after its input

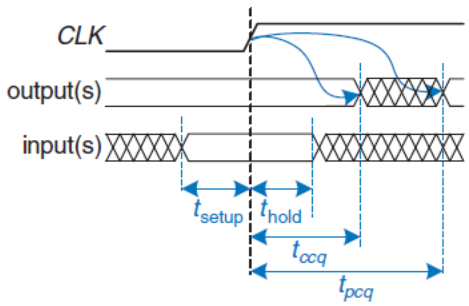

Figure 3.37 Timing specification for synchronous sequential circuit

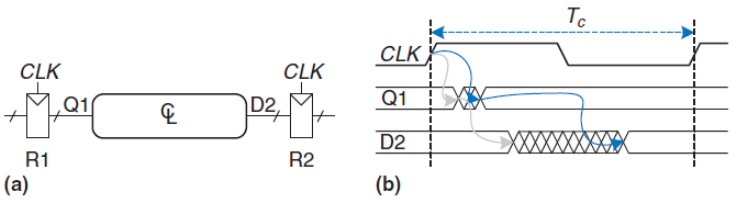

Figure 3.38 Path between registers and timing diagram

changes and settles to the final value within a propagation delay after its input settles. The gray arrows represent the contamination delay through R1 and the combinational logic, and the blue arrows represent the propagation delay through R1 and the combinational logic. We analyze the timing constraints with respect to the setup and hold time of the second register, R2.

Setup Time Constraint

Figure 3.39 is the timing diagram showing only the maximum delay through the path, indicated by the blue arrows. To satisfy the setup time of R2, *D*2 must settle no later than the setup time before the next clock edge. Hence, we find an equation for the minimum clock period:

$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} \tag{3.13}$$

In commercial designs, the clock period is often dictated by the Director of Engineering or by the marketing department (to ensure a competitive product). Moreover, the flip-flop clock-to-Q propagation delay and setup time, t_{pcq} and t_{setup}, are specified by the manufacturer. Hence, we rearrange Equation 3.13 to solve for the maximum propagation delay through the combinational logic, which is usually the only variable under the control of the individual designer.

$$t_{pd} \leq T_c - (t_{pcq} + t_{\text{setup}}) \tag{3.14}$$

The term in parentheses, $t_{pcq} + t_{\text{setup}}$, is called the *sequencing overhead*. Ideally, the entire cycle time T_c would be available for useful

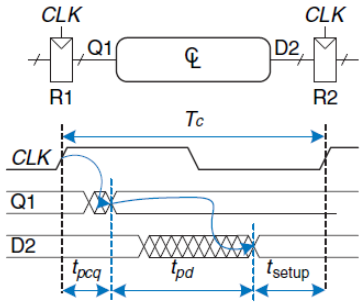

Figure 3.39 Maximum delay for setup time constraint

computation in the combinational logic, t_{pd}. However, the sequencing overhead of the flip-flop cuts into this time. Equation 3.14 is called the *setup time constraint* or *max-delay constraint*, because it depends on the setup time and limits the maximum delay through combinational logic.

If the propagation delay through the combinational logic is too great, $D2$ may not have settled to its final value by the time R2 needs it to be stable and samples it. Hence, R2 may sample an incorrect result or even an illegal logic level, a level in the forbidden region. In such a case, the circuit will malfunction. The problem can be solved by increasing the clock period or by redesigning the combinational logic to have a shorter propagation delay.

Hold Time Constraint

The register R2 in Figure 3.38(a) also has a *hold time constraint*. Its input, $D2$, must not change until some time, t_{hold}, after the rising edge of the clock. According to Figure 3.40, $D2$ might change as soon as $t_{ccq} + t_{cd}$ after the rising edge of the clock. Hence, we find

$$t_{ccq} + t_{cd} \geq t_{\text{hold}} \tag{3.15}$$

Again, t_{ccq} and t_{hold} are characteristics of the flip-flop that are usually outside the designer's control. Rearranging, we can solve for the minimum contamination delay through the combinational logic:

$$t_{cd} \geq t_{\text{hold}} - t_{ccq} \tag{3.16}$$

Equation 3.16 is also called the *hold time constraint* or *min-delay constraint* because it limits the minimum delay through combinational logic.

We have assumed that any logic elements can be connected to each other without introducing timing problems. In particular, we would expect that two flip-flops may be directly cascaded as in Figure 3.41 without causing hold time problems.

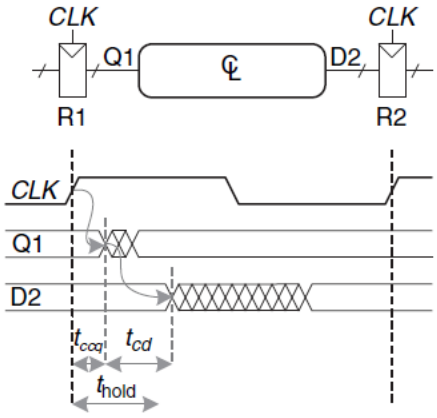

Figure 3.40 **Minimum delay for hold time constraint**

In such a case, $t_{cd} = 0$ because there is no combinational logic between flip-flops. Substituting into Equation 3.16 yields the requirement that

$$t_{\text{hold}} \leq t_{ccq} \tag{3.17}$$

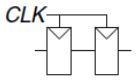

Figure 3.41 Back-to-back flip-flops

In other words, a reliable flip-flop must have a hold time shorter than its contamination delay. Often, flip-flops are designed with $t_{\text{hold}} = 0$, so that Equation 3.17 is always satisfied. Unless noted otherwise, we will usually make that assumption and ignore the hold time constraint in this book.

Nevertheless, hold time constraints are critically important. If they are violated, the only solution is to increase the contamination delay through the logic, which requires redesigning the circuit. Unlike setup time constraints, they cannot be fixed by adjusting the clock period. Redesigning an integrated circuit and manufacturing the corrected design takes months and millions of dollars in today's advanced technologies, so *hold time violations* must be taken extremely seriously.

Putting It All Together

Sequential circuits have setup and hold time constraints that dictate the maximum and minimum delays of the combinational logic between flip-flops. Modern flip-flops are usually designed so that the minimum delay through the combinational logic is 0—that is, flip-flops can be placed back-to-back. The maximum delay constraint limits the number of consecutive gates on the critical path of a high-speed circuit, because a high clock frequency means a short clock period.

Example 3.10 TIMING ANALYSIS

Ben Bitdiddle designed the circuit in Figure 3.42. According to the data sheets for the components he is using, flip-flops have a clock-to-Q contamination delay of 30 ps and a propagation delay of 80 ps. They have a setup time of 50 ps and a hold time of 60 ps. Each logic gate has a propagation delay of 40 ps and a

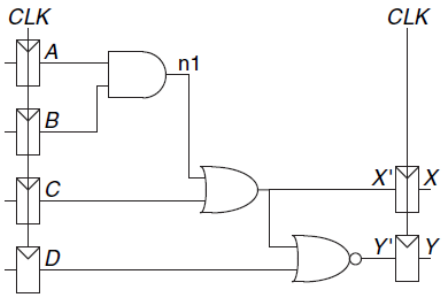

Figure 3.42 Sample circuit for timing analysis

contamination delay of 25 ps. Help Ben determine the maximum clock frequency and whether any hold time violations could occur. This process is called *timing analysis*.

Solution: Figure 3.43(a) shows waveforms illustrating when the signals might change. The inputs, A to D, are registered, so they only change shortly after CLK rises.

The critical path occurs when $B = 1$, $C = 0$, $D = 0$, and A rises from 0 to 1, triggering n1 to rise, X' to rise, and Y' to fall, as shown in Figure 3.43(b). This path involves three gate delays. For the critical path, we assume that each gate requires its full propagation delay. Y' must setup before the next rising edge of the CLK. Hence, the minimum cycle time is

$$T_c \geq t_{pcq} + 3\,t_{pd} + t_{\text{setup}} = 80 + 3\times 40 + 50 = 250\text{ps} \qquad (3.18)$$

The maximum clock frequency is $f_c = 1/T_c = 4$ GHz.

A short path occurs when $A = 0$ and C rises, causing X' to rise, as shown in Figure 3.43(c). For the short path, we assume that each gate switches after only a contamination delay. This path involves only one gate delay, so it may occur after $t_{ccq} + t_{cd} = 30 + 25 = 55$ ps. But recall that the flip-flop has a hold time of

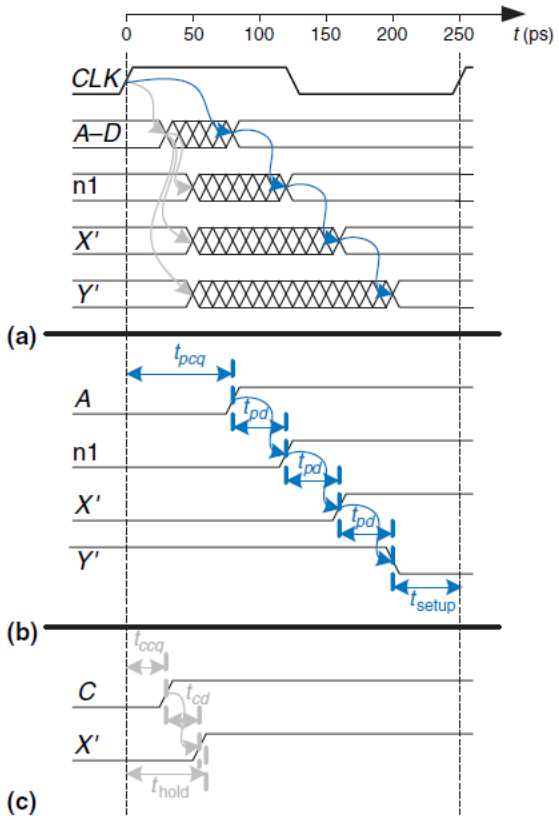

Figure 3.43 **Timing diagram: (a) general case, (b) critical path, (c) short path**

60 ps, meaning that X' must remain stable for 60 ps after the rising edge of CLK for the flip-flop to reliably sample its value. In this case, $X' = 0$ at the first rising edge of CLK, so we want the flip-flop to capture $X = 0$. Because X' did not hold stable long enough, the actual value of X is unpredictable. The circuit has a hold time violation and may behave erratically at any clock frequency.

Example 3.11 FIXING HOLD TIME VIOLATIONS

Alyssa P. Hacker proposes to fix Ben's circuit by adding buffers to slow down the short paths, as shown in Figure 3.44. The buffers have the same delays as other gates. Determine the maximum clock frequency and whether any hold time problems could occur.

Solution: Figure 3.45 shows waveforms illustrating when the signals might change. The critical path from A to Y is unaffected, because it does not pass through any buffers. Therefore, the maximum clock frequency is still 4 GHz. However, the short paths are slowed by the contamination delay of the buffer. Now X' will not change until $t_{ccq} + 2t_{cd} = 30 + 2 \times 25 = 80$ ps. This is after the 60 ps hold time has elapsed, so the circuit now operates correctly.

This example had an unusually long hold time to illustrate the point of hold time problems. Most flip-flops are designed with $t_{\text{hold}} < t_{ccq}$ to avoid such problems.

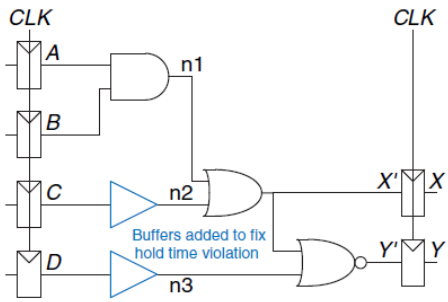

Figure 3.44 Corrected circuit to fix hold time problem

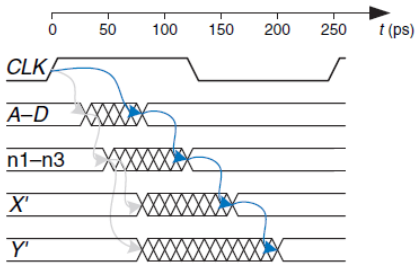

Figure 3.45 Timing diagram with buffers to fix hold time problem

However, some high-performance microprocessors, including the Pentium 4, use an element called a *pulsed latch* in place of a flip-flop. The pulsed latch behaves like a flip-flop but has a short clock-to-Q delay and a long hold time. In general, adding buffers can usually, but not always, solve hold time problems without slowing the critical path.

3.5.3 Clock Skew*

In the previous analysis, we assumed that the clock reaches all registers at exactly the same time. In reality, there is some variation in this time. This variation in clock edges is called *clock skew*. For example, the wires from the clock source to different registers may be of different lengths, resulting in slightly different delays, as shown in Figure 3.46. Noise also results in different delays. Clock gating, described in Section 3.2.5, further delays the clock. If some clocks are gated and others are not, there will be substantial skew between the gated and ungated clocks. In Figure 3.46, *CLK*2 is *early* with respect to *CLK*1, because the clock wire between the two registers follows a scenic route. If the clock had been routed differently, *CLK*1 might have been early instead. When doing timing analysis, we consider the worst-case scenario, so that we can guarantee that the circuit will work under all circumstances.

Figure 3.47 adds skew to the timing diagram from Figure 3.38. The heavy clock line indicates the latest time at which the clock signal might reach any register; the hashed lines show that the clock might arrive up to t_{skew} earlier.

First, consider the setup time constraint shown in Figure 3.48. In the worst case, R1 receives the latest skewed clock and R2 receives the earliest skewed clock, leaving as little time as possible for data to propagate between the registers.

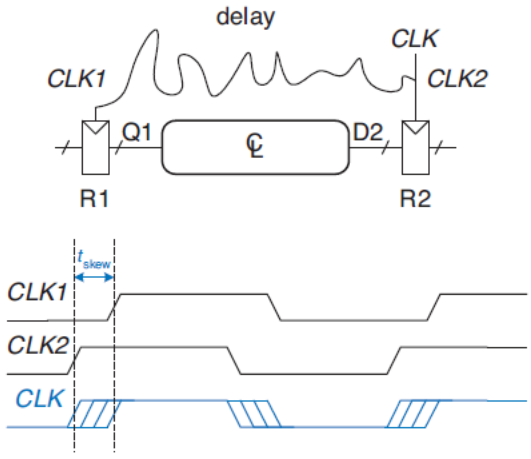

Figure 3.46 Clock skew caused by wire delay

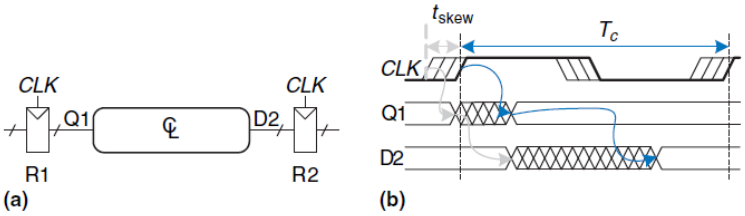

Figure 3.47 Timing diagram with clock skew

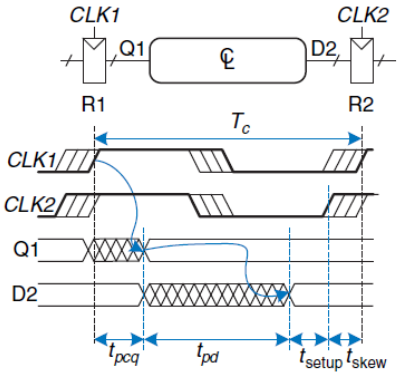

Figure 3.48 Setup time constraint with clock skew

The data propagates through the register and combinational logic and must setup before R2 samples it. Hence, we conclude that

$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} + t_{\text{skew}} \tag{3.19}$$

$$t_{pd} \leq T_c - (t_{pcq} + t_{\text{setup}} + t_{\text{skew}}) \tag{3.20}$$

Next, consider the hold time constraint shown in Figure 3.49. In the worst case, R1 receives an early skewed clock, *CLK*1, and R2 receives a late skewed clock, *CLK*2. The data zips through the register and combinational logic but must not arrive until a hold time after the late clock. Thus, we find that

$$t_{ccq} + t_{cd} \geq t_{\text{hold}} + t_{\text{skew}} \tag{3.21}$$

$$t_{cd} \geq t_{\text{hold}} + t_{\text{skew}} - t_{ccq} \tag{3.22}$$

In summary, clock skew effectively increases both the setup time and the hold time. It adds to the sequencing overhead, reducing the time available for useful work in the combinational logic. It also increases the required minimum delay through the combinational logic. Even if $t_{\text{hold}} = 0$, a pair of back-to-back flip-flops will violate Equation 3.22 if $t_{\text{skew}} > t_{ccq}$. To prevent

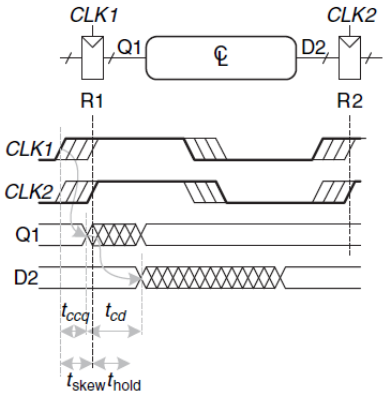

Figure 3.49 Hold time constraint with clock skew

serious hold time failures, designers must not permit too much clock skew. Sometimes flip-flops are intentionally designed to be particularly slow (i.e., large t_{ccq}), to prevent hold time problems even when the clock skew is substantial.

Example 3.12 TIMING ANALYSIS WITH CLOCK SKEW

Revisit Example 3.10 and assume that the system has 50 ps of clock skew.

Solution: The critical path remains the same, but the setup time is effectively increased by the skew. Hence, the minimum cycle time is

$$\begin{aligned} T_c &\geq t_{pcq} + 3t_{pd} + t_{\text{setup}} + t_{\text{skew}} \\ &= 80 + 3\times 40 + 50 + 50 = 300\text{ps} \end{aligned} \tag{3.23}$$

The maximum clock frequency is $f_c = 1/T_c = 3.33$ GHz.

The short path also remains the same at 55 ps. The hold time is effectively increased by the skew to $60 + 50 = 110$ ps, which is much greater than 55 ps. Hence, the circuit will violate the hold time and malfunction at any frequency. The circuit violated the hold time constraint even without skew. Skew in the system just makes the violation worse.

Example 3.13 FIXING HOLD TIME VIOLATIONS

Revisit Example 3.11 and assume that the system has 50 ps of clock skew.

Solution: The critical path is unaffected, so the maximum clock frequency remains 3.33 GHz.

The short path increases to 80 ps. This is still less than $t_{hold} + t_{skew} = 110$ ps, so the circuit still violates its hold time constraint.

To fix the problem, even more buffers could be inserted. Buffers would need to be added on the critical path as well, reducing the clock frequency. Alternatively, a better flip-flop with a shorter hold time might be used.

3.5.4 Metastability

As noted earlier, it is not always possible to guarantee that the input to a sequential circuit is stable during the aperture time, especially when the input arrives from the external world. Consider a button connected to the input of a flip-flop, as shown in Figure 3.50. When the button is not pressed, $D = 0$. When the button is pressed, $D = 1$. A monkey presses the button at some random time relative to the rising edge of *CLK*. We want to know the output Q after the rising edge of *CLK*. In Case I, when the button is pressed much before *CLK*, $Q = 1$. In Case II, when the button is not pressed until long after *CLK*, $Q = 0$. But in Case III, when the button is pressed sometime between t_{setup} before *CLK* and t_{hold} after *CLK*, the input violates the dynamic discipline and the output is undefined.

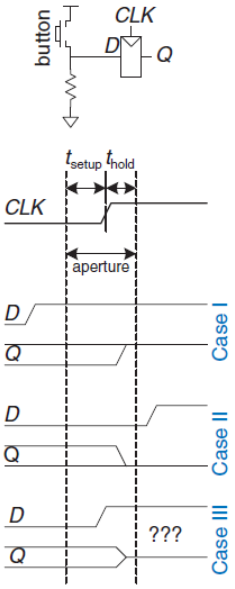

Figure 3.50 Input changing before, after, or during aperture

Metastable State

When a flip-flop samples an input that is changing during its aperture, the output Q may momentarily take on a voltage between 0 and V_{DD} that is in the forbidden zone. This is called a *metastable* state. Eventually, the flip-flop will resolve the output to a *stable state* of either 0 or 1. However, the *resolution time* required to reach the stable state is unbounded.

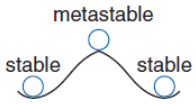

Figure 3.51 Stable and metastable states

The metastable state of a flip-flop is analogous to a ball on the summit of a hill between two valleys, as shown in Figure 3.51. The two valleys are stable states, because a ball in the valley will remain there as long as it is not disturbed. The top of the hill is called metastable because the ball would remain there if it were perfectly balanced. But because nothing is perfect, the ball will eventually roll to one side or the other. The time required for this change to occur depends on how nearly well balanced the ball originally was. Every bistable device has a metastable state between the two stable states.

Resolution Time

If a flip-flop input changes at a random time during the clock cycle, the resolution time, t_{res}, required to resolve to a stable state is also a random variable. If the input changes outside the aperture, then $t_{res} = t_{pcq}$. But if the input happens to change within the aperture, t_{res} can be substantially longer. Theoretical and experimental analyses (see Section 3.5.6) have

shown that the probability that the resolution time, t_{res}, exceeds some arbitrary time, t, decreases exponentially with t:

$$P(t_{res} > t) = \frac{T_0}{T_c} e^{-\frac{t}{\tau}} \tag{3.24}$$

where T_c is the clock period, and T_0 and τ are characteristic of the flip-flop. The equation is valid only for t substantially longer than t_{pcq}.

Intuitively, T_0/T_c describes the probability that the input changes at a bad time (i.e., during the aperture time); this probability decreases with the cycle time, T_c. τ is a time constant indicating how fast the flip-flop moves away from the metastable state; it is related to the delay through the cross-coupled gates in the flip-flop.

In summary, if the input to a bistable device such as a flip-flop changes during the aperture time, the output may take on a metastable value for some time before resolving to a stable 0 or 1. The amount of time required to resolve is unbounded, because for any finite time, t, the probability that the flip-flop is still metastable is nonzero. However, this probability drops off exponentially as t increases. Therefore, if we wait long enough, much longer than t_{pcq}, we can expect with exceedingly high probability that the flip-flop will reach a valid logic level.