High Speed CMOS VLSI Design

Lecture 9: Sequential Logic I

1.0 Introduction to Sequential Logic Design

So far, all logic we have considered computes functions of the current inputs. Real systems must have memory so that they can also compute functions of previous inputs. Such functions are implemented with *sequential* logic.

A major application of sequential logic is for pipelined processor design. Logic moves through stages; for example, the output of the Decode stage becomes the input of the Execute stage. By dedicating one cycle to each stage, multiple instructions can simultaneously occupy the pipeline so that one is being executed while the next is being decoded. On each cycle, the instructions advance to the next stage.

Another application of sequential logic is for finite state machines. On each step, the state machine takes inputs and its current state and produces outputs and advances to the next state.

There are several ways to implement memory elements; they can be categorized as static and dynamic storage. Both rely on the capacitance of the storage node to hold energy which indicates the state. Dynamic storage just places charge on the node. Dynamic storage must be refreshed periodically so that it does not leak away. Static storage uses positive feedback to actively restore the energy; hence static memory retains its contents indefinitely.

It is important to remember that the choice of static or dynamic storage elements is distinct from the choice of static or dynamic logic. For example, static storage elements can be used in a pipeline built out of domino gates. Alternately, dynamic storage elements could be used in a finite state machine constructed with static gates.

In this lecture, we will explore sequential design techniques with application to static logic. The three major options for storage elements are edge-triggered flip-flops, transparent latches, and pulsed latches. We will see how each kind of storage element adds overhead which increases the cycle time. Then we will look at actual circuit designs of various elements. We'll finish with special topics in sequential design, including scan circuits, pulse generators and clock choppers, and wave pipelining.

In the next lecture, we'll move on to sequential design techniques applicable to dynamic logic. We'll see that good design methodologies are essential to minimizing the sequential

overhead of dynamic logic. We'll also see that the quality of the clock signal is very important for high-performance design, so in the third lecture we will consider techniques for generating and distributing clocks or for eliminating the clock altogether with asynchronous design.

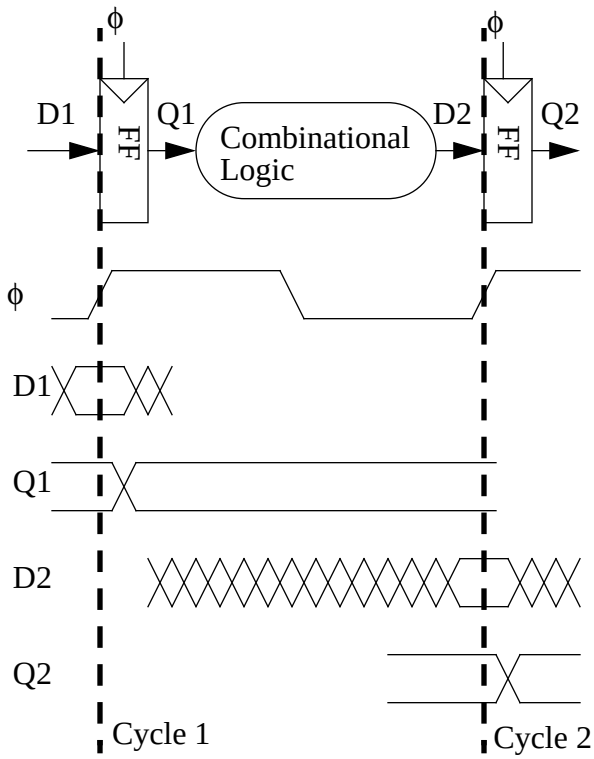
2.0 Sequential Elements

Although there are a multitude of possible sequential elements including S-R latches and J-K flip-flops, most CMOS systems are built with just three types of sequential elements: edge-triggered flip-flops, transparent latches, and pulsed latches. There is some confusion about terminology in the industry, so this section will try to be careful with the names of sequential elements. All three sequential elements have clock and data inputs and an output. Depending on the design, the output may use true, complementary, or both polarities.

The three elements are described below. Each has a timing diagram underneath. The logic is drawn along the horizontal axis corresponding to the time at which the data arrives.

Edge-triggered flip-flops are also known as master-slave or D flip-flops. When the clock rises, the data input is sampled and transferred to the output. At all other times, the data input and output are unrelated.

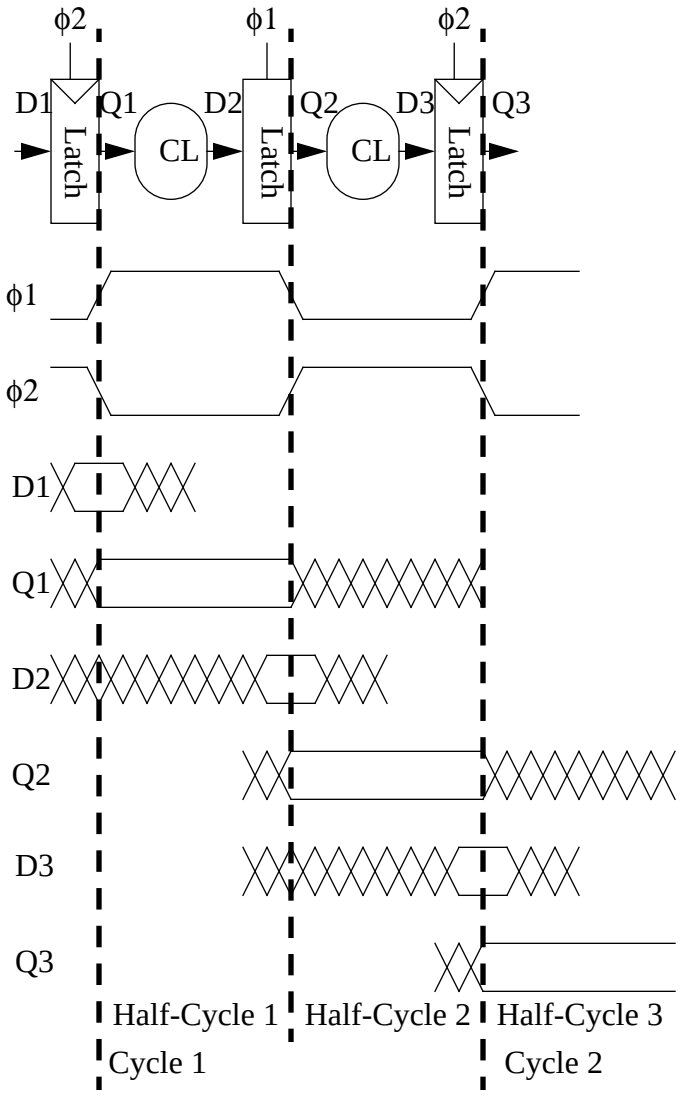
FIGURE 1. Flip-flop System



Transparent latches are also known as half-latches. When the clock is high, the output follows the input. When the clock is low, the output holds at its last value and ceases to track

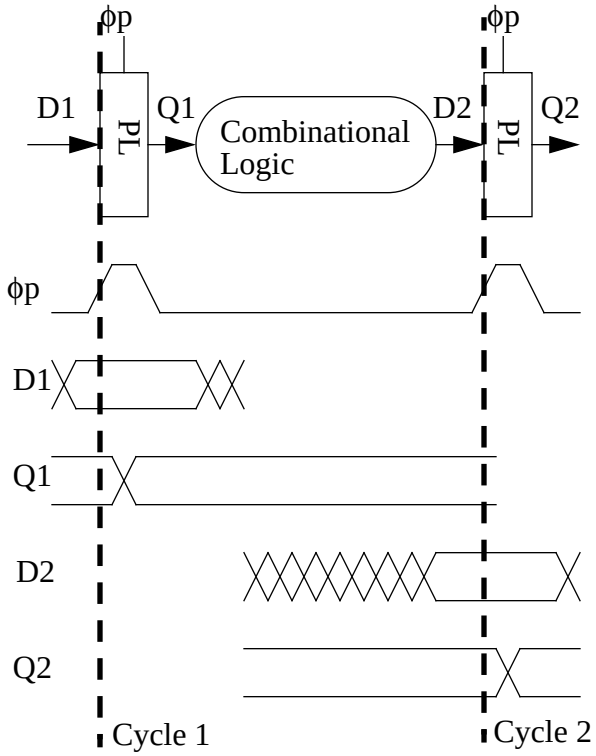
the input. It is confusing to call a latch ON or OFF because it is not clear whether ON means that latch is passing data or ON means that latch is holding old data. Instead, the terms transparent and opaque are clearer. To build a sequential system, two half-latches must be used in each cycle. One is transparent for the first part of the cycle, while the other is transparent for the second part of the cycle.

FIGURE 2. Transparent Latch System



Pulsed latches behave exactly as transparent latches, but instead of controlling them with a 50% duty cycle clock, they are given a short pulse. As long as the pulse is shorter than the time required for data to propagate through the logic before the next pulsed latch, only one pulsed latch is required in each cycle. Pulsed latches are often misnamed as edge-triggered flip-flops, but they are actually quite distinct.

FIGURE 3. Pulsed Latch System



3.0 Cycle Time of Sequential Systems

Ideally, the cycle time of a system should equal the propagation delay through the longest logic path. Unfortunately, real sequential systems introduce overhead from many sources that increases the cycle time. This section lists the various sources of overhead and shows how much overhead applies to each of the systems described above. Based on the analysis, it recommends the best latching techniques for various applications.

3.1 Sequential Overhead Sources

The most obvious source of overhead from sequential elements is the propagation delay through the element. This depends on the kind of element and the circuit design.

Another source is setup time. This is the time that the data input must have settled before the clock input arrives.

Clock skew is the variation in arrival time of the clock. It may hurt performance because data should be ready by the earliest the clock may arrive, yet the sequential element output may not change until the latest the clock arrives. Sources of clock skew will be discussed later. We will draw clock waveforms at the latest they may arrive; actual clocks may arrive up to $t_{ske}w$ earlier.

Time borrowing is the ability for critical logic that nominally operates in one cycle or half-cycle to use time left by non-critical logic in an adjacent half-cycle or cycle. Some sequential elements support time borrowing, while others do not. Latching methods which support time borrowing have numerous advantages. They are easier to design because logic can be balanced between cycles or half-cycles. If there is insufficient time at the end of a cycle for an entire gate, systems using time borrowing can apply the remaining time to logic in another cycle, while systems without time borrowing waste the time. Finally, time borrowing helps compensate for inaccuracies in modeling or variations in processing that make some cycles longer and some shorter than predicted because the long cycles can borrow some of the slack from the short cycles.

In addition to these sources of overhead, all synchronous systems operating off a fixed frequency clock pay environmental and manufacturing margins. Manufacturing margins occur because a part is only sold at specific frequencies; for example, if a chip is sold in 800 MHz and 1 GHz speed bins, a part that operates at up to 970 MHz must be sold in the 800 MHz bin. Environmental margins occur because the chip must be able to operate at its rated frequency under worst case voltage and temperature, while in an actual system the true environment is likely to be better. These margins can be reduced if the clock frequency can be arbitrarily adjusted. Manufacturing margin can be minimized by selling the chip at exactly its worst case operating frequency. Environmental margin can also be reduced by measuring on-chip temperature and voltage and adaptively changing the frequency. All of these techniques require accurate knowledge of the worst-case path and good tracking with the worst-case path delay, which can be a difficult problem.

3.2 Analysis of Latching Techniques

Manufacturing and environmental margins apply to all latching techniques, so will not be considered further. Let’s examine how other sources of overhead impact cycle time.

Flip-flops require the data input to settle a setup time before the earliest a skewed clock might arrive. The output becomes valid a propagation delay after the latest a skewed clock might arrive. Therefore, the cycle time is:

$$t_{cycle} = t_{logic} + t_{d->q} + t_{setup} + t_{skew} \quad \text{(EQ 1)}$$

Transparent latches are better because they can use time borrowing to hide overhead. Data must be designed to arrive at the latch $t_{setup} + t_{skew}$ before the falling edge of the latch clock. However, this time is not wasted; instead the next half-cycle can begin using the result as soon as it arrives. Thus, in a system constructed entirely of 2-phase transparent latches:

$$t_{cycle} = t_{logic} + t_{d1->q1} + t_{d2->q2} \quad \text{(EQ 2)}$$

where $t_{d1->q1}$ and $t_{d2->q2}$ are the delays of the two static latches. Remember that a flip-flop is just two back-to-back latches, so the propagation delay of a flip-flop is comparable to the propagation delay of two latches.

Also, additional time borrowing is useful in transparent latch systems to balance logic between phases and compensate for uncertainty in delays.

Pulsed latch systems have pros and cons relative to transparent latches. The biggest advantage is that now only a single latch is used, reducing the propagation delay. The data must arrive at least $t_{setup} + t_{skew}$ before the falling edge of the pulse. If the pulse is wide enough, the pulsed latch will be transparent when the data arrives and there is no wasted time. If the pulse is narrower than this window, there is extra overhead. A wide pulse is also beneficial to allow limited amounts of time borrowing. Unfortunately, a wide pulse also increases the number of min-delay problems in which short paths must be padded. The cycle time of a pulsed-latch system is:

$$t_{cycle} = t_{logic} + t_{d->q} + \max\{0, t_{setup} + t_{skew} - t_{pulsewidth}\} \quad \text{(EQ 3)}$$

3.3 Recommendations

Flip-flops were once popular because they are conceptually simplest. As cycle times are getting shorter, the overhead of flip-flops is too large and they have gone out of favor. Thus, the two main contenders are transparent latches and pulsed latches.

Transparent latches can eliminate all overhead except their propagation delay. They also allow nearly half a cycle of time borrowing. Pulsed latches can be even faster because only a single latch delay is introduced. However, hiding the setup time and clock skew requires a relatively wide pulse. Such a pulse leads to min-delay problems, as will be seen. Pulsed latches also provide very little time borrowing.

Designers willing to do extensive min-delay checking are beginning to favor pulsed latches. Other designers prefer the safety and time borrowing of transparent latches. Fast systems can be built with either technique.

4.0 Min-Delay

If there is little logic in a cycle, the clock starting the cycle is skewed early, and the clock ending the cycle is skewed late, data can race through the cycle and be clocked into the next clocked element all in one cycle. This is bad.

The contamination delay through logic in a flop-based system must be at least:

$$t_{cd} = t_{skew} + t_{hold} - t_{cd\text{-}flop} \quad \text{(EQ 4)}$$

The min-delay constraints on 2-phase logic depend how much the phases may overlap. Academic designs often use non-overlapping clock phases to eliminate min-delay problems. Unfortunately, this requires either distributing two phases around the entire chip or building complex clock generators, so high-performance industrial designs usually clock the logic with true and complementary versions of a global clock with nominal 50% duty cycle. For such systems, the contamination delay constraint is the same as for flop-based systems:

$$t_{cd} = t_{skew} + t_{hold} - t_{cd\text{-}latch1} - t_{cd\text{-}latch2} \quad \text{(EQ 5)}$$

Pulsed latch systems have the most difficult contamination delay restrictions:

$$t_{cd} = t_{skew} + t_{hold} + t_{pulse\text{-}wdith} - t_{cd\text{-}latch} \quad \text{(EQ 6)}$$

In principle, min-delay problems should only occur on paths with a small amount of logic. Min-delay can be avoided by padding the path with extra inverters or with an extra pulsed latch operating in the middle of the cycle.

In practice, some paths have widely differing min and max delays. The most difficult ones to manage are those which are critical in the worst case, but so fast in the best case that they could cause min-delay failure.

5.0 Latch Circuits

There are many ways to build latches and flip-flops. A flip-flop can just be constructed from two back-to-back transparent latches. A pulsed latch is just a transparent latch controlled by a pulse rather than a 50% duty cycle clock. Thus, the most important building block is the transparent latch.

The simplest dynamic latch is just a pass transistor, shown in Figure 4. To provide rail-to-rail output swings, a full transmission gate is usually a better choice.

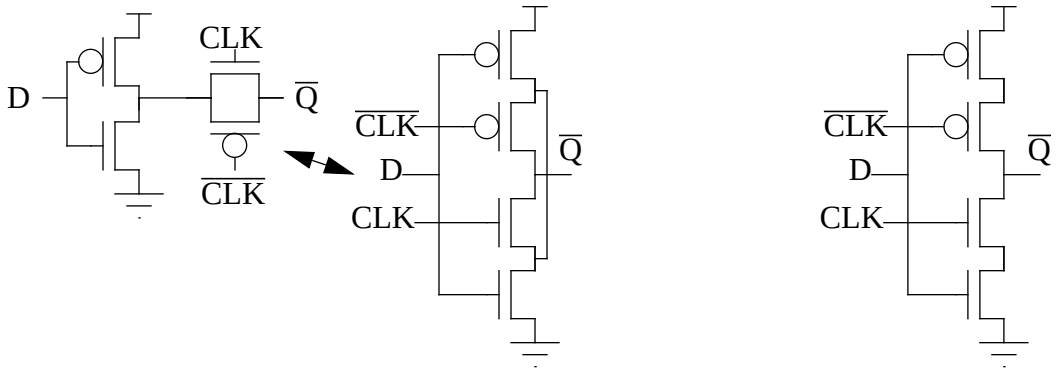
FIGURE 4. Simple Latches

Pass-Transistor Latch Transmission Gate Latch

Such a latch must drive only a capacitive load; it cannot drive diffusion inputs of other pass transistors because charge sharing could cause the output to glitch when the latch is opaque. Coupling onto the output node is also a problem because it is a dynamic node. Latch setup time depends on both the load and driving gate, just as the delay through an ordinary transmission gate depends on the driver and load.

To solve these problems, inverters may be placed on the input, output, or both. When an inverter is placed on the input, sometimes the inputs to the transmission gate are separated to create a tri-state inverter instead, as shown in Figure 5. The performance of both designs is comparable; removing the shorting connection slightly reduces drive capability, but also reduces internal diffusion parasitics.

FIGURE 5. Inverter + Transmission Gate vs. Tri-state



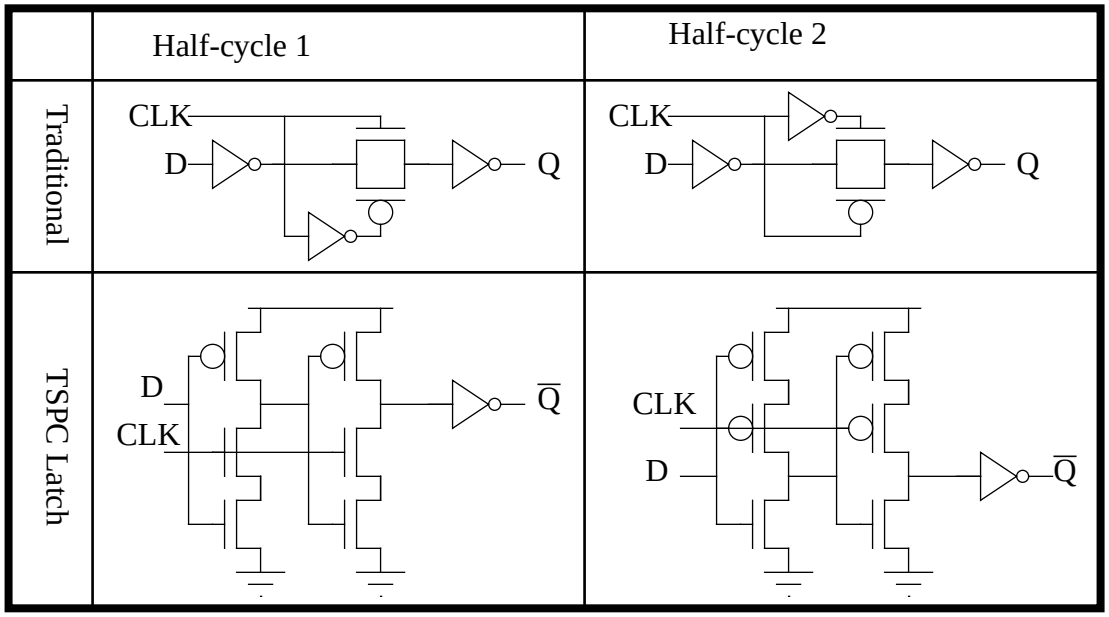
Equivalent Circuits of Inverter + Transmission Gate

Tristate

Note that it is important for the clock input to be on the inside of the stack. If the data input were on the inside, changes in the data value while the latch was off could cause charge sharing and glitches on the output.

To minimize clock skew, most chips distribute only a single clock signal across the entire chip. Locally, additional phases are derived as necessary. For example, an inverter may be used to create $\overline{CLK}$ from CLK. Once upon a time, some designer felt that generating local phases was also bad for skew. The True Single-Phase Clocking (TSPC) techniques enjoyed a period of popularity and were actually used on the Alpha 21064. They are shown below:

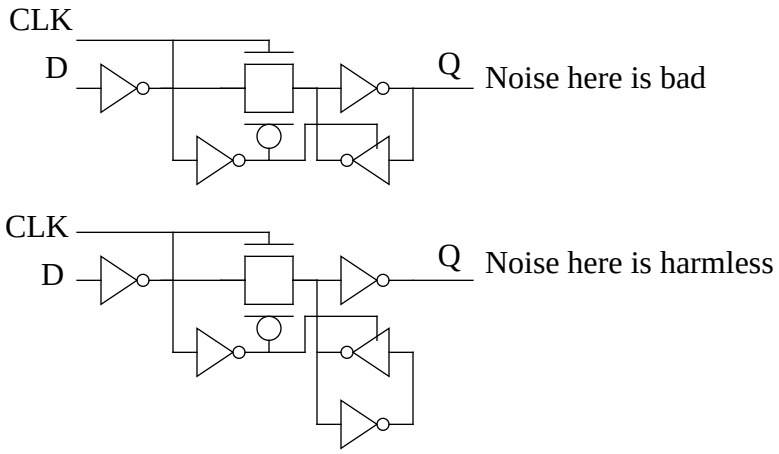
FIGURE 6. TSPC vs. Traditional Latches



TSPC latches have a longer propagation delay because the data passes through more gates. Since inverting the clock can be done relatively cheaply and TSPC latches are slower and larger than normal latches, TSPC is seldom used anymore.

All latches presented so far are dynamic; if left indefinitely, they may leak to an incorrect value. To allow a system to run at low frequency or with a stopped clock, the dynamic node must be staticized. This can be done with a weak feedback inverter or with a tri-state feedback inverter. Weak inverters introduce a ratio problem and must be weak enough to be overwritten by the input driver. Tristate inverters are thus safer and avoid contention power. Feeding back from the output inverter is risky because noise on the output can overwrite the contents of the latch. Thus, the output is often buffered with a separate gate, as shown in Figure 7.

FIGURE 7. Staticizing Latches



For highest performance custom design, a bare transmission gate is excellent and was used by DEC in the Alpha 21164. Care should be taken with the driver and load that setup time can be met. For synthesized design, an inverter/transmission gate/inverter combination latch is safer to guarantee setup times and safety of the dynamic node. In any case, designs which must support stop clock or low-frequency operation should staticize the dynamic node with cross-coupled weak inverters.