CHAPTER 5
Sequential Logic design practices

This chapter describes the most commonly used and dependable sequential-circuit design methods carried out on synchronous systems. Synchronous systems are those in which all flip-flops are clocked by the same, common clock signal.

This chapter begins with a quick summary of sequential circuit documentation standards. This is followed by the most basic building blocks of sequential-circuit design—latches, flip-flops, counters and shift registers.

5.1 Sequential Circuit documentation standards.

- State-machine layout. Within a logic diagram, a collection of flip-flops and combinational logic that forms a state machine should be drawn in a logical format on the same page.

- Cascaded elements. In a similar way, registers, counters and shift registers that use multiple IC’s should have the multiple IC’s grouped together in the schematic so that the cascading structure is obvious.

- Flip-Flops. The symbols for individual sequential-circuit elements should follow drawing standards.

- State-machine descriptions. State machines should be described by state tables, state diagrams, transition lists, ASM charts, or test files in a state machine description language.

- Timing diagrams. The documentation package for sequential circuits should include timing diagrams that show the general timing assumptions and timing behavior of the circuit.

- Timing specifications A sequential circuit should be accompanied by a specification of the timing requirements for proper internal operation (e.g., maximum clock frequency), as well as the requirements for any externally supplied inputs (e.g., setup and hold-time requirements with respect to the system clock, minimum pulse widths, etc).

5.2 SSI Latches and Flip-flops

Several different types of discrete latches and flip-flops are available as SSI parts. These devices are most commonly used in the design of state machines and “unstructured” sequential circuits that don’t fall into the category of shift registers, counters and other sequential MSI functions. However, SSI latches and flip-flops are eliminated to a large extent in modern designs as their functions are embedded in larger,
structured logic devices such as PLDs. Nevertheless some of them are still used in most
digital systems.

Some of the most common SSI parts are 74x74 which contains two independent
positive-edge-triggered D flip-flops with preset and clear inputs. Faster version of this
part namely 74F74 and 74ACT74, find application in synchronizers for asynchronous
input signals.

5.3 Multi-bit Registers and Latches

A collection of two or more D flip-flops with a common clock input is called a
register. Registers are often used to store a collection of related bits, such as a byte of
date in a computer. However a single register can also be used to store unrelated bits of
data or control information; the only real constraint is that all of the bits are stored using
the same clock signal.

Many digital systems including computers, telecommunications devices, and
stereo equipment, process 16 or 8 bits of data at a time; as a result, IC’s that handle eight
bits are very popular. One such MSI IC is the 74x374 octal edge-triggered D flip-flop
also simply known as 8-bit register is shown in fig 5.1
Figure 5.1 Logic diagram for 74x374 8-bit register with its pin numbers.
Fig 5.2 Logic symbol for 74x374 8-bit register.

Fig 5.3 Logic symbol for 74x373 8-bit latch

One variation of 74x374 is the 74x373. The ‘373 uses D latches instead of edge-triggered flip-flops. Therefore its outputs follow the corresponding inputs whenever C is asserted, and latch the last input when C is negated.
5.3.1 Counters

The name counter is used for any clocked sequential circuit whose state diagram contains a single cycle, as shown in fig 5.4. The modulus of a counter is the number of states in the cycle. A counter with m states is also called a modulo-m counter or sometimes, a divide-by-m counter. A counter whose modulus is not a power of 2 will, of necessity contains extra states that are not used in normal operation.

The most commonly used counter type is n-bit binary counter. Such a counter has n flip-flops and has $2^n$ states, which are visited in the sequence 0, 1, and 2,..., $2^n-1$, 0, 1. Each of the foregoing states is encoded in the corresponding n-bit binary integer.

![Fig 5.4 General Structure of a counter’s state diagram-a single cycle.](image)

5.3.2 Ripple Counters

An n-bit binary ripple counter is constructed using a set of T flip-flops. Each bit of this counter toggle if and only if the immediately preceding bit changes from 1 to 0. This corresponds to a normal binary counting sequence-when particular bit changes from 1 to 0; it generates a carry to the most significant bit. The counter is called a ripple counter as the carry information from the less significant bit ripples from the less significant bits to the more significant bits one at a time. This is shown in figure 5.5.
5.3.3 Synchronous counters

Synchronous counter Fig 5.6 connects all of its flip-flop clock inputs to the same common clock CLK signal, so that all of the flip-flop output change at the same time after only $t_{pq}$ ns of delay where $t_{pq}$ is the propagation delay of each toggle flip-flop. Like the binary ripple counter, a synchronous n-bit binary counter can be built with a fixed amount of logic per bit-in this case, a T flip-flop with enable and a 2-input AND gate.

![Diagram of a 4-bit binary ripple counter](image)

Figure 5.5 A 4-bit binary ripple counter.

This is also called synchronous serial counter because the combinational enable signals propagate serially from the least significant bit to the most significant bits. If the clock period is too short, there may not be enough change for a change in the counter’s LSB to propagate to MSB. Driving each enable input with a dedicated AND gate, just a single level of logic eliminates this problem. Called a synchronous parallel counter, this is the fastest binary counter structure.
5.3.4 MSI Counters.

The most popular counter is the 74x163, a synchronous 4-bit binary counter with load and clear inputs. The ‘163 use D flip-flops rather than T flip-flops internally to facilitate the load and clear functions. Each D input is driven by 2-input multiplexer consisting of an OR gate and two AND gates.

5.3.5 Shift Registers

A shift register is an n-bit register with a provision for shifting its stored data by one bit position at each tick of the clock. Figure 5.8 shows the structure of serial-in, serial-out shift register. The serial input, SERIN, specifies a new bit to be shifted into one end at each clock tick. This bit appears at the serial output, SEROUT, after n clock ticks, and is lost one tick later. Thus an n-bit serial-in serial-out shift register can be used to delay a signal by n clock ticks.

A serial-in parallel-out register is shown in figure 5.9, has outputs for all its stored bits, making them available to other circuits. Such a shift register can be used to perform serial-to-parallel conversion.
5.3.6 MSI Shift registers

The two most popular MSI 8-bit shift registers are 74x164 a serial-in parallel out device with an asynchronous clear input and 74x166 a parallel-in, serial-out shift register with a asynchronous clear input. The registers described so far are unidirectional shift registers because they can shift its contents in only one direction. 74x194 is a bi-directional shift register. It is capable of shifting data in both directions. 74x299 is a universal shift register because it can be made to function unidirectional register or bi-directional register. The most common application of shift registers is to convert parallel data into serial format for transmission or storage, and to convert serial data back to parallel format for processing or display.

5.3.7 Shift-Register Counters.
A shift register can be combined with combinational logic to form a state machine whose state diagram is cyclic. Such a circuit is called a shift-register counter. Unlike a binary counter a shift-register counter does not count in an ascending or descending sequence binary sequence, but is useful in many “control” applications. Some of the shift register counters are ring counters, Johnson counters, Linear Feedback Shift register counters.

5.4 Impediments To synchronous design.

5.4.1 Clock Skew.

Synchronous systems using edge-triggered flip-flops work properly only if all flip-flops see the triggering clock edge at the same time. Consider two flip-flops FF1 and FF2 clocked by the same signal, but the clock signal seen by FF-2 is delayed by significant amount of time relative to FF1’s clock. This difference between arrival times of the clock at different devices is called a clock skew.

It can be determined if clock skew is a problem in this case by defining \( t_{skew} \) to be the amount of clock skew and using other timing parameters like flip-flop FF2 hold time, combinatorial delay, FF1 propagation delay and hold time of flip-flop FF2. For proper operation

\[
T_{f_{p_{d(min)}}} + t_{c_{omb(min)}} - t_{h_{old}} - t_{skew(max)} > 0
\]

In other words clock skew subtracts from the hold time margin. The example given here can happen in several ways. In a large system a single clock may not have adequate fanout to drive all the devices with clock inputs, so it maybe necessary to provide two or more copies of the clock signal. One method of doing this is by buffering the clock as shown in figure 5.10 and fig5.11.

![Figure 5.10 Buffering the clock thereby introducing excessive clock skew](image)
The buffering method of figure 5.10 produces excessive clock skew since CLOCK1 and CLOCK2 are delayed through an extra buffer compared to CLOCK. A recommended buffering method is shown in figure 5.11. The entire clock signals in this case through identical buffers and has roughly equal delays. Ideally all the buffers should be part of the same IC package, so that they all have similar delay characteristics and are operating at identical power supply voltage and temperature.

Even the method in Figure 5.11 produces excessive clock skew if one clock signal is loaded more heavily than the other; transitions on the more heavily loaded clock appear to occur later because of increases in output transistor switching delay and signal rise and fall times. Therefore it becomes necessary for a designer to balance the loads on multiple clocks, looking at both DC load (fanout) and AC load (wiring and input capacitance).

Another bad situation can occur when signals on a PCB or in an ASIC are routed automatically by a CAD program. Under this condition clock skew is introduced due to improper layout of clock signals by the CAD program. This clock skew problem may be detected using CAD timing analysis program. Then the problem can be remedied by inserting extra delay in the too-fast data paths.

**5.4.2 Gating the clock.**

A risky design method used to enable or disable the clock signal is called gating a clock. This is shown in Fig 5.12. A signal CLKEN is asserted to enable the clock, and is simply ANDed with the clock to produce the gated clock GCLK. This approach has two problems:

![Figure 5.12 Improper way of gating the clock](image)

If CLKEN is a state-machine output or other signal produced by a register clocked by CLOCK, then CLKEN changes some time after CLOCK has already gone
HIGH. This produces glitch on GCLK, and false clocking at the registers controlled by GCLK.

Even if CLKEN is somehow produced well in advance of CLOCK’s rising edge (e.g., a register clocked with the falling edge CLOCK) the AND-gate delay gives GCLK excessive clock skew, which causes more problems all around.

A method of gating the clock that generates, only minimal clock skew is shown in Figure 5.13. Here, both an ungated clock and several gated clocks are generated from the same active-low master clock signal. Gates in the same IC package are used to minimize the possible differences in their delays. The CLKEN signal may change arbitrarily whenever /CLOCK is LOW, which is when CLOCK is HIGH. This condition is okay as CLKEN signal is typically produced by a state machine whose outputs change right after CLOCK goes HIGH.

5.4.3 Asynchronous Inputs

Asynchronous inputs are typically request for service (e.g., interrupts in a computer system) or status flags. The transition rates are asynchronous signals may range from less than one per second to 1MHz or more. Under these circumstances synchronizer is used to produce an output that meets the setup and hold times required in a synchronous system. Though synchronizer may introduce metastability by careful design of the synchronizer the metastability problem may be reduced.

![Figure 5.13 An acceptable way to gate the clock](image.png)