5. Sequential CMOS Logic Circuits

In sequential logic circuits the output signals is determined by the current inputs as well as the previously applied input variables. Fig. 5.1a shows a sequential circuit consisting of a combinational circuit and a memory block in the feedback loop. In most cases, the regenerative behaviour of sequential circuits is due to either a direct or an indirect feedback connection between the output and the input. Regenerative operation can, under certain conditions, also be interpreted as a simple memory function. Regenerative circuits can be classified into three main groups: bistable circuits, monostable circuits, and astable circuits. The general classification of non-regenerative and regenerative logic circuits is shown in Fig. 5.1.

Bistable states have two stable states or operation modes, each of which can be attained under certain input and output conditions. Monostable circuits have only one stable operating point (state). Astable circuits have no stable operating point or state which the circuit can preserve for a certain time period. Among these main groups of regenerative circuit types, the bistable circuits are by far the most widely used and most important class. All basic latches flip-flops, registers and memory elements used in digital systems fall into this category.

Figure 5.1 (a) Sequential circuit consisting of a combinational logic block and a memory block in a feedback loop (b) Classification of logic circuits based on their temporal behaviour
Behaviour of Bistable Elements

Fig. 5.2a shows an example of a bistable circuit which consists of two identical cross-coupled inverter circuits. Here the output voltage of inverter (1) is equal to the input voltage of inverter (2), (i.e. $v_{o1} = v_{i2}$), and the output voltage of inverter (2) is equal to the input voltage of inverter (1), (i.e. $v_{o2} = v_{i1}$). The voltage transfer characteristic of inverter (1) and (2) are plotted in Fig. 5.2b. Notice that the input and output voltages of inverter (2) correspond to the output and input voltages of inverter (1), respectively.

It can be seen that the two voltage transfer characteristics intersect at three points. The stable operating points of the circuit are labelled in Fig 5.2b. If the circuit is operating at one of these two stable points, it will preserve this state unless it is forced externally to change its operating point. Note that the gain of each inverter circuit, i.e., the slope of the respective voltage transfer curves, is smaller than unity at the two stable operating points. Thus a significantly large external voltage perturbation must be applied so that the voltage gain of the inverter loop becomes larger than unity to change the state of the circuit.

Note if the circuit is biased at the unstable point, a small voltage perturbation at the input of any of the inverters will be amplified (since gain of the unstable biasing point is greater than unity) causing the operating point to move to one of the stable operating points.

---

**Figure 5.2** Static behaviour of the two inverter basic bistable circuit: (a) Circuit schematic (b) Intersecting voltage transfer curves of the two inverter, showing the three possible operating points. (c) Qualitative view of the potential energy levels corresponding to the three operating points
The bistable behaviour of the cross-coupled inverter circuit can also be studied by examining the total potential energy levels at each of the three possible operating points (Fig. 5.2c). It can be seen that the potential energy is at its minimum at the two stable operating points, since the voltage gains of both inverters are equal to zero. The energy attains a maximum at the unstable operating point since the voltage gains of both inverters are its maximum.

Fig. 5.3a) shows the circuit diagram of a CMOS two-inverter bistable element. Note that at the unstable operating point of this circuit, all four transistors are in saturation, resulting in the maximum loop gain for the circuit. If the initial operating condition is set at this point, any small voltage perturbation will cause significant changes in the operating modes of the transistors. Thus the output voltages of the two inverters will eventually diverge to $V_{OH}$ and $V_{OL}$, respectively, as illustrated in Fig 5.3b. The direction in which the output voltage diverges depends on the initial perturbation polarity.

![Circuit diagram of a CMOS bistable circuit](image)

**Figure 5.3** (a) Circuit diagram of a CMOS bistable circuit. (b) One possibility for the expected time domain behaviour of the output voltages, if the circuit is initially set at its unstable operating point.

**SR Latch Circuit (using NOR gates)**

The simple two-inverter circuit alone has no provision for allowing its state to be changed externally from one stable operating model to the other. To allow such a change of state, we must add simple switches to the bistable circuit, which can be used to force or trigger the circuit from one operating point to the other. Fig 5.4 shows the circuit structure of the simple CMOS SR latch, which has two such triggering inputs, S (set) and R (reset). In the literature, the SR latch is also called the SR flip-flop, since two stable states can be switched back and forth. The circuit consists of two CMOS NOR2 gates. The gate-level schematic of the SR latch is shown in Fig 5.5.

![Gate-level schematic of the SR latch](image)

The SR latch circuit has two complementary outputs, $Q$ and $\bar{Q}$. By definition, the latch is said to be in its set state when $Q$ is logic “1” and $\bar{Q}$ is logic “0” (Inputs S="1" and R="0"). Conversely, the latch is in its reset state when $Q$ is logic “0” and $\bar{Q}$ is logic “1” (Inputs S="0" and R="1"). When both inputs S and R are set to logic “0”, the SR latch
Figure 5.4 CMOS SR latch circuit based on NOR2 gate

Figure 5.5 Gate-level schematic and block diagram of the NOR-based SR latch.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
<td>( Q_{n+1} )</td>
<td>( \bar{Q}_{n+1} )</td>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( Q_n )</td>
<td>( \bar{Q}_n )</td>
<td>Hold</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Set</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reset</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Not allowed</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1 Truth table of the NOR-based SR latch circuit

will operate exactly like the simple cross-coupled bistable circuit (Fig. 5.2) and will hold either one of its two stable operating points (states) as determined by the previous inputs.

The truth table of the NOR-based SR latch is summarized in Table 5.1. Note when both inputs S and R are equal to logic “1”, the output nodes will be forced to logic “0” which conflicts with the complementary of \( Q \) and \( \bar{Q} \). Therefore, this input combination is not permitted during normal operation and is considered to be a not allowed condition.

The operation of the CMOS SR latch circuit of Fig. 5.4 can be examined in more detail by considering the operating modes of the four nMOS transistors, M1, M2, M3 and M4. If the set input (S) is equal to \( V_{OH} \) and the reset input (R) is equal to \( V_{OL} \), both the parallel connected transistors M1 and M2 will be on. Consequently, the voltage on node \( \bar{Q} \) will become a logic-low level of \( V_{OL} = 0 \). At the same time, both M3 and M4 are turned off,
which results in a logic high voltage $V_{OH}$ at node $Q$. If the reset (R) is equal to $V_{OH}$ and the set input (S) is equal to $V_{OL}$, the situation will be reversed (M1 and M2 turned off and M3 and M4 turned on). When both of the input voltages are equal to $V_{OL}$, there are two possibilities depending on the previous state. Either M2 or M3 will be on, while both of the trigger transistors M1 and M4 are off.

For transient analysis of the SR latch circuit is examined when a state changes. In this scenario, both the output nodes undergo simultaneous voltage transitions. Thus an interesting problem is to estimate the amount of time required for the simultaneous switching of the two output nodes. The exact solution of this problem requires solving two coupled differential equations, one for each output node. The problem can be simplified considerably if you assume that the two events take place in sequence rather than simultaneously. This assumption causes an overestimation of the switching time.

To calculate the switching times for both output nodes, the total parasitic capacitance associated with each node must be calculated as

$$
C_Q = C_{gb,2} + C_{gb,5} + C_{db,3} + C_{db,4} + C_{db,7} + C_{sb,7} + C_{db,8}
$$
$$
C_{\bar{Q}} = C_{gb,3} + C_{gb,7} + C_{db,1} + C_{db,2} + C_{db,5} + C_{sb,5} + C_{db,6}
$$

The circuit diagram of the SR latch is shown in Fig. 5.6 together with the lumped load capacitances at nodes $Q$ and $\bar{Q}$. Assuming that the latch is initially reset an that a set operation is being performed by applying S="1" and R="0", the rise time associated with node Q can now be estimated as

$$
\tau_{rise,Q}(SR \ latch) = \tau_{rise,Q}(NOR2) + \tau_{fall,\bar{Q}}(NOR2)
$$

Figure 5.6 Circuit diagram of the CMOS SR latch showing the lumped load capacitances at both output nodes.
Note that (5.2) overestimates the actual switching time for the SR latch since the switching events are calculated sequentially rather than simultaneously. This approach, however, yields a simpler first order prediction for the time delay, as opposed to the simultaneous solution of two coupled differential equations.

**SR Latch Circuit (using NAND gates)**

Fig. 5.7 shows a SR latch using two NAND2 gates. Note in order to hold (preserve) a state, both of the external trigger inputs must be equal to logic “1”. The state of the circuit can be changed by pulling the set input or reset input to zero. The gate level schematic and the corresponding block diagram representation of the NAND-based SR latch are shown in Fig. 5.8.

![CMOS SR latch circuit based on NAND2 gates](image)

*Figure 5.7 CMOS SR latch circuit based on NAND2 gates*

![Gate-level schematic and block diagram of the NAND based SR latch.](image)

*Figure 5.8 Gate-level schematic and block diagram of the NAND based SR latch.*
Note that a NAND-based SR latch responds to an active low input signals, while the NOR-based SR latch responds to an active high inputs. The small circles at the S and R input terminals indicate that the circuit responds to active low input signals. The truth table of the NAND SR latch is also shown in Fig. 5.8. The same approach used in the timing analysis of the NOR-based SR latches can be applied to NAND-based SR latches.

**Clocked Latch and Flip-Flop Circuits**

**Clocked SR Latch**

The SR latches examined in the previous sections are essentially asynchronous sequential circuits, which respond to the changes occurring in input signals at a circuit-delay-dependent time point during operation. To facilitate synchronous operation, the circuit response can be controlled simply by adding a gating clock signal to the circuit, so that the outputs will respond to the input levels only during the active period of the clock pulse.

The gate-level schematic of a clocked NOR-based SR latch is shown in Fig. 5.9. Note if the clock (CK) is equal to logic “0”, the input signals have no influence upon the circuit response and the SR latch will hold its current state. When the clock input goes to logic “1”, the logic levels applied to the S and R inputs are permitted to reach the SR latch. To illustrate the operation of the clocked SR latch, the waveforms of Fig. 5.10 are provided. Note that the circuit is strictly level-sensitive during active phases (i.e. any changes occurring in the S and R input voltages when CK level is equal to logic “1” will be reflected onto the circuit outputs. Consequently, even a narrow spike or glitch occurring during an active clock phase can set or reset the latch, if the loop delay is shorter than the pulse width.

Fig. 5.11 shows a CMOS implementation of a clocked NOR-based SR latch circuit, using two simple AOI (AND-OR-INVERT) gates. Notice that the AOI-based implementation of the circuit results in a very small transistor count, compared with the alternative circuit realization consisting of two AND2 and two NOR2 gates.

![Figure 5.9 Gate-level schematic of the clocked NOR-based latch.](image)
The NAND-based SR latch can also be implemented by gating the clock input, as shown in Fig. 5.12. For the implementation of Fig. 5.12, both the input signals S and R as well as the clock signal CK are active low. This means that in the input signal will be ignored when the clock is equal to logic “1” and that inputs will influence the outputs only when the clock is active (i.e. CK=“0”). For the circuit implementation of this clocked NAND-based SR latch, an OAI (OR-AND-INVERT) structure can be used, which is analogous to the AOI-based realization of the clocked NOR SR latch.

A different implementation of the clocked NAND-based SR latch is shown in Fig. 5.13. Here, both input signals and the CK signal are active high. Fig. 5.13 behaves as Fig 5.9. The drawback of this implementation is that the transistor count is higher than the active low version shown in Fig. 8.12.
Figure 5.12  Gate-level schematic of the clocked NAND-based SR latch circuit, with active low inputs.

Figure 5.13  (a) Gate-level schematic of the clocked NAND-based SR latch circuit, with active high inputs. (b) Partial block diagram representation of the same circuit.

Figure 5.14  (a) Gate-level schematic of the clocked NAND-based JK latch circuit

Clocked JK Latch

All simple and clocked SR latch circuits examined to this point suffer from the common problem of having a not-allowed input combination (i.e. their state becomes indeterminate when both S and R are activated at the same time. This problem can be overcome by adding two feedback lines from the outputs to the inputs, resulting in the JK latch (shown is Fig 5.14). Fig. 5.15 shows an all NAND implementation of the JK latch with active high inputs, and the corresponding block diagram representation.
Figure 5.15 All-NAND implementation of the clocked JK latch circuit

Table 5.2 Detailed truth table of the JK latch circuit

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>$Q_n$</th>
<th>$\overline{Q}_n$</th>
<th>S</th>
<th>R</th>
<th>$Q_{n+1}$</th>
<th>$\overline{Q}_{n+1}$</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Hold</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reset</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Set</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Toggle</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.16 (a) Gate-level schematic of the clocked NOR-based JK latch circuit (b) CMOS AOI realization of the JK latch
The J and K inputs in this circuit correspond to the set and reset inputs of the basic SR latch. When the clock is active, the latch can be set with the input combination \((J=\text{"1"}, K=\text{"0"})\), and it can be reset with the input combination \((J=\text{"0"}, K=\text{"1"})\). If both inputs are equal to logic “0”, the latch preserves its current state. If both inputs are equal to logic “1” during the active clock phase, the latch simply switches its state due to the feedback. Thus, the JK latch does not have a not-allowed input combination. As in other clocked latch circuits, the JK latch will hold its current state when \(CK=\text{"0"}\). The operation of the clocked JK latch is summarized in Table 5.2.

Fig. 5.16 shows an alternative NOR-based implementation of the clocked JK latch, and the CMOS realization using the AOI circuit structure.

While there is not-allowed input combination for the JK latch, there is still a potential problem. If both inputs are equal to logic “1” during the active phase of the clock pulse, the output of the circuit will oscillate (toggle) continuously until either clock becomes inactive (\(CK\) goes to zero), or one of the input signals goes to zero. To prevent this undesirable timing problem, the clock pulse width must be made smaller than the input-to-output propagation delay of the JK latch circuit (thus the clock signal must go low before the output has an opportunity to switch again, which prevents uncontrolled oscillation). This clock constraint is difficult to implement for most practical applications.

**Master-Slave Flip-Flop**

Fig. 5.17 shows two cascaded latches activated with opposite clock phases. This configuration is referred to as a master-slave flip-flop configuration. The input latch in Fig. 5.17 is called the “master” and is activated when the clock pulse is high. During this phase, the inputs J and K allow data to be entered into the flip-flop. When the clock pulse goes to zero, the master latch becomes inactive and the second-stage latch, called the “slave” becomes active. The output levels of the flip-flop circuit are determined during this second phase, based on the master-stage outputs set in the previous phase.

Note, a change occurring in the primary inputs is never reflected directly to the outputs. This important property clearly separates the master-slave flip-flop from all of the latch circuits examined earlier. Fig. 5.18 shows a sample set of input and output waveforms associated with the JK master-slave flip-flop.

![Master-slave flip-flop consisting of NAND based JK latch](image-url)

**Figure 5.17** Master-slave flip-flop consisting of NAND based JK latch
Figure 5.18 Sample input and output waveforms of the master-slave flip-flop circuit

Note, the master and the slave stages are decoupled from each other, the circuit allows for toggling when \( J = K = 1 \), but it eliminates the possibility of uncontrolled oscillations since only one stage is active at any given time. Fig. 5.18 also shows that the master-slave flip-flop has the potential problem of “ones catching”. When the clock pulse is high, a narrow spike or glitch in one of the inputs, for instance a glitch in the J line (or K line), may set (or reset) the master latch and thus cause an unwanted state transition, which will
then be propagated into the slave stage during the following phase. This problem can be eliminated to a large extent by building an edge-triggered master-slave flip-flop. Fig. 5.19 shows an alternative NOR-based realization of a master-slave flip-flop circuit.

**CMOS D-Latch and Edge-Triggered Flip-Flop**

In general, direct CMOS implementations of conventional circuits such as clocked JK latch or the JK master-slave flip-flop tend to require a large number of transistors. This section examines specific versions of sequential circuits built primarily with CMOS transmission gates since they generally are simpler and require fewer transistors than the circuits designed with conventional structuring.

As an example, consider the simple D-latch circuit shown in Fig. 5.20. The gate level representation of the D-latch is simply obtained by modifying the clocked NOR-based SR latch circuit. It can be seen from Fig. 5.20 that the output Q assumes the value of the input D when the clock is active (i.e. for CK=“1”). When the clock signal goes to zero, the output will simply preserve its state. Thus, the CK input acts as an enable signal which allows data to be accepted into the D-latch.

The transistor implementation of a D-latch circuit can be realized as shown in Fig. 5.21. Fig 5.21 shows a basic two-inverter loop and two CMOS transmission gate (TG) switches. The CMOS transmission gate (TG) consists of one nMOS and one pMOS transistor connected in parallel and acts as a bidirectional switch between nodes A and B which is controlled by signal C, as illustrated by Fig. 5.22.

The TG at the input is activated by the CK signal, whereas the TG in the inverter loop is activated by the inverse of the CK signal, (CK)’. Thus, the input signal is accepted (latched) into the circuit when the clock is high, and this information is preserved as the state of the inverter loop when the clock is low. The operation of the CMOS D-latch circuit can be better visualized by replacing the CMOS transmission gates with simple switches, as shown in Fig. 5.23. The timing diagram (Fig.5.23) shows the time intervals during which the input and the output signals should be valid (unshaded).

Note that the valid D input must be stable for a short time before (setup time, $t_{\text{setup}}$) and after (hold time, $t_{\text{hold}}$) the negative clock transition, during which the input switch opens and the loop switch closes. Note that any violation of the setup and hold times can cause metastability problems which can lead to unpredictable transient behaviour.

![Figure 5.20 Gate-level schematic and the block diagram view of D-latch](image-url)
Figure 5.21 CMOS implementation of the D-latch (version 1)

Figure 5.22 Four different representations of the CMOS transmission gate (TG)

Figure 5.23 Simplified schematic and timing diagram, showing the setup and hold times.
Fig. 5.24 shows a different version of the CMOS D-latch. The circuit contains two tristate inverters, driven by the clock signal and its inverse. The operation of Fig 5.24 is similar to Fig. 5.21. The first tri-state inverter acts as the input switch, accepting the input signal when the clock is high. At this time, the second tristate inverter is at its high-impedance state, and the output Q is following the input signal. When the clock goes low, the input buffer becomes inactive, and the second tristate inverter completes the two-inverter loop, which preserves its state until the next clock pulse.

The D-latches of Fig. 5.21 and Fig. 5.24 are not an edge-triggered storage element because the output changes according to the input (i.e. the latch is transparent, while clock is high). Fig. 5.25 shows the two-stage master-slave flip-flop circuit constructed by two D-latch circuits. The first stage (master) is driven by the clock signal, while the second stage (slave) is driven by the inverted clock signal. Thus the master stage is a positive level-sensitive, while the slave stage is negative level-sensitive.
Fig. 5.26 shows sampled input and output waveforms of the CMOS negative edge-triggered D-type flip-flop. The output of the master stage latches the applied input (D) when the clock signal is “1” and the output of the slave state becomes valid when the clock signal drops to logic “0”. Thus the D-type flip-flop (DFF) essentially samples in the input at every falling edge of the clock pulse.

Note that the DFF circuit can be seriously affected if the master stage experiences a setup or hold time violation. An example of a setup violation is illustrated in Fig. 5.27, where the input D switches from “0” to “1” immediately before the clock transition occurs (setup time violation). As a result, the master stage fails to latch the correct value, and the slave stage produces an erroneous output. The relative timing of the input and clock signals must be carefully synchronized to avoid such situations. The layout of the CMOS DFF circuit is given in Fig. 5.27.

Figure 5.26 Simulated input and output waveforms of the CMOS DFF circuit (Fig. 5.25)
Figure 5.27 Simulated waveforms of the CMOS DFF circuit, showing a set-up time violation for the master stage input at 10ns. The output of the master stage fails to settle at the correct level.

Figure 5.28 Layout of the CMOS DFF shown in Fig. 5.25
Another implementation of edge-triggered D flip-flop is shown in Fig 5.29, which consists of six NAND3 gates. This DFF is positive edge-triggered as illustrated in Fig. 5.30.

![Figure 5.29 NAND3-based positive edge-triggered D flip-flop circuit](image1)

**Figure 5.29** NAND3-based positive edge-triggered D flip-flop circuit

![Figure 5.30 Timing diagram of the positive edge-triggered D flip-flop](image2)

**Figure 5.30** Timing diagram of the positive edge-triggered D flip-flop