SEQUENTIAL CIRCUITS

- In the first part of the course, **combinational circuits** were covered.
  The outputs of combinational circuits depend only on **current** inputs.
  Combinational circuit: Output = G (Input)

- In **sequential circuits**, the outputs depend both on the inputs and the "state" of the circuit.
  Sequential circuit: Output = G (Input , Current State)
  Next State = H (Input , Current State)

Memory units are required to store (remember) the state of the circuit.

For example, a vending machine keeps track of (remembers) the coins inserted into the machine.

With each coin, the state of the machine (the total amount of money corresponding to the inserted coins) is updated.

Types of sequential circuits:

There are two types of sequential circuits :

A) Synchronous sequential circuits:

- Their states can change at a discrete instant of time.
- All memory elements are synchronized by a common **clock signal**.
- Therefore, these circuits are also called "clocked synchronous sequential" circuits.

B) Asynchronous sequential circuit:

- Their state can change at any instant of time depending upon the input signals.

In this course, we will deal only with clocked synchronous sequential circuits because nearly all sequential logic today is clocked synchronous.

For example, **microprocessors** are clocked synchronous sequential circuits.

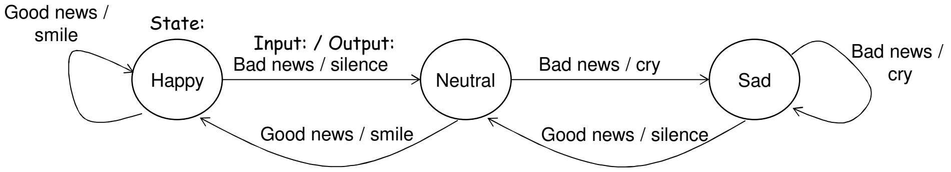
Finite State Machine (FSM) Model

Sequential circuits are designed using the “finite state machine – FSM” model.
This model is also used in the design of many other systems.
An FSM has inputs, states, and outputs.

- When the machine is started, it is in a specific state (initial state: s_0).
- An output is produced depending on the inputs and the current state. O = G (I,S)
- Transition into a new state occurs depending on the input and the current state.

To illustrate the behavior of an FSM, state/output diagrams are used.

Example: A state/output diagram for an FSM that models the behavior of a human

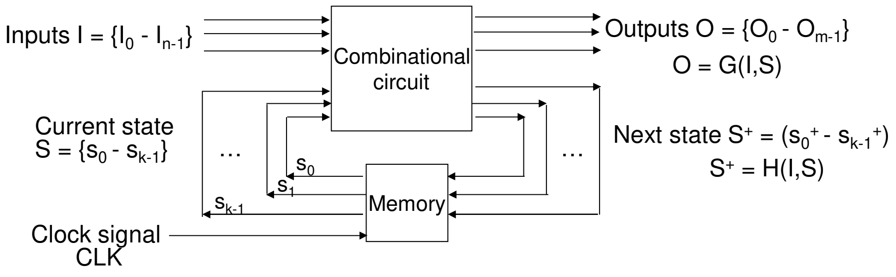


Finite State Machine (FSM) Model (cont.)

A digital circuit designed using the FSM model has two parts:

a) Combinational circuit for logic operations to determine output and next state
b) Memory unit to remember the current state

Block diagram of a clocked synchronous sequential circuit:


Inputs I = {I_0 - I_{n-1}}
Combinational circuit
Outputs O = {O_0 - O_{m-1}}
O = G(I,S)
Current state S = {s_0 - s_{k-1}}
…
s_0
s_1
s_{k-1}
Memory
…
Next state S^+ = (s_0^+ - s_{k-1}^+)
S^+ = H(I,S)
Clock signal CLK

We will see details of the FSM and clocked synchronous sequential circuits (in chapters 9 and 10) after we cover memory units.

Memory Units

Latches and Flip-flops:

A basic memory cell is a circuit that stores **one bit** of information for as long as the device is powered.

This one-bit memory element is called a flip-flop or a latch because it latches (or locks) data in it.

A memory element that has no clock input is often called a **latch**.

It is not triggered by a control signal (i.e., a clock signal).

The value of the latch can be changed whenever the latch is enabled.

A **flip-flop** is a memory unit that is triggered by a **clock signal**.

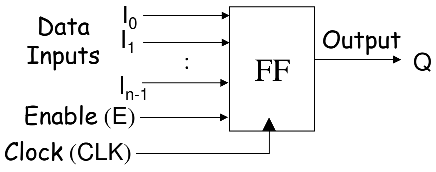
In this chapter, we will cover the details of latches and flip-flops.

Latches and Flip-flops (cont'd):

There are different types of latches and flip-flops.

Depending on its type, a latch or a flip-flop has one or more data inputs, a single output, and control inputs.

Example: A general flip-flop:



Characteristic function of the flip-flop:
$Q(t^+) = f(\,Q(t), I_0, I_1, \ldots, I_{n-1}\,)$

$Q(t)$: Current value
$Q(t^+)$: Next value

The **Q output** shows the current value (state) of the memory unit (0,1).

The next value of the output Q (denoted by $Q(t+1)$, $Q(t^+)$, or Q^+) is a function of the current state (denoted by $Q(t)$ or Q) and the current inputs.

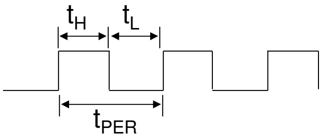
The **clock signal** (denoted by CLK) determines when the next state function is evaluated, and the output of the flip-flop changes its value.

The output of the flip-flop can only change when the clock signal is active (the definition of being active will be described in the coming slides).

If CLK is not active, the flip-flop output will not change even if input values change.

Clock Signal:

The clock signal is a periodic square wave that synchronizes the gates in the circuit.



A logic unit that has a clock signal input (i.e., flip-flop) is enabled only when the clock signal is active. If the clock signal is not active, the flip-flop preserves its state.

There are two types of units that differ in how they use the clock signal:

a) Level-triggered units b) Edge-triggered units

Level-triggered units use a level of the clock signal (1 in positive logic) as active.

A level-triggered unit becomes active and changes its output when the clock signal is at the **high** level.

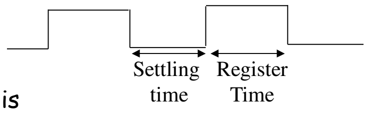
It preserves its state when the clock signal is at the **low** level.

When the clock signal is at the high level ("1"), the inputs should not change as they are being processed.

Otherwise, the output of the sequential circuit is indeterminate (random).

This time is called the **register time**.

The inputs can change when the clock signal is 0. This time is called the **settling time**.

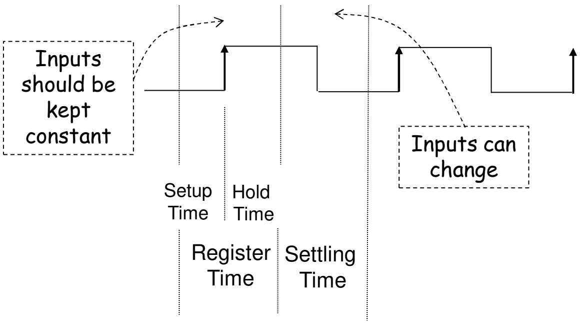


Edge-triggered units:

These units use an edge (rising edge in positive logic) of the clock signal as active.

A positive edge-triggered unit uses the 0→1 transition of the clock signal (rising-edge) to change its state and output. At other times, it preserves its state.

As the inputs are used (processed) during a 0→1 (or 1→0)transition, inputs should be kept constant for a certain amount of time before and after the transition. Otherwise, the output of the sequential unit is indeterminate (random).



Inputs should be kept constant

Inputs can change

Setup Time

Hold Time

Register Time

Settling Time

The **register time** is the sum of the setup and hold times.

Setup time is the minimum time the data signal should be steady **before** the clock transition.

Hold time is the minimum time the data signal should be steady **after** the clock transition.

The inputs should be kept constant during the register time so that the sequential circuit works correctly.

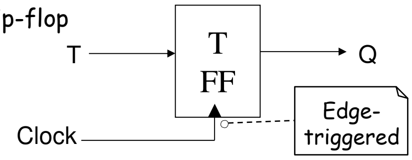
In **negative logic**, this processing occurs at the **1→0 transition (falling edge)**.

Example: Edge-Triggered Toggle (or Trigger) (T) Flip-Flop

There are four main types of latches and flip-flops: S-R, D, T, and J-K.

The major difference in these flip-flop types is their characteristic function.

Example: T flip-flop

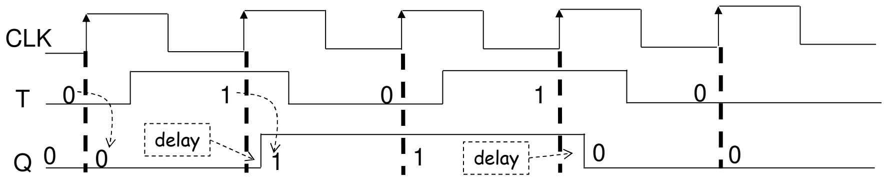


Characteristic Equation:
$Q(t^+)= T \oplus Q(t)$

The next output (state) of the T flip-flop $Q(t^+)$ is equal to its current output (state) XORed with its input (T).

According to this, the output of the flip-flop does not change when T=0 because
$0 \oplus x = x$.

The output of the flip-flop is inverted (complemented) when T=1 because
$1 \oplus x = \bar{x}$.

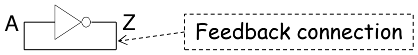


Feedback Connections

The combinational circuits we have studied so far have not had feedback connections. To construct a circuit that has memory, we must introduce feedback into the circuit.

By **feedback**, we mean that the output of one of the gates is connected back to the input of another gate in the circuit to form a closed loop.

Example:

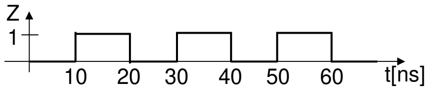


If, at some instant of time, the inverter input A is "0", this "0" will propagate through the inverter and cause the output Z to become "1".

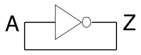
This "1" is fed back into the input, so after the propagation delay, the inverter output Z will become "0".

When this 0 feeds back into the input A, the output Z will again switch to "1", and so forth.

Inverter output Z will continue to oscillate back and forth between "0" and "1" as shown in the figure below, and it will never reach a stable condition (it will remain **unstable**).

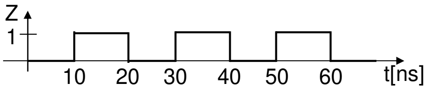


Feedback Connections (cont'd)



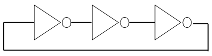
This **unstable** circuit cannot be used as a memory unit.

However, this circuit can be used to measure the propagation delay of the inverter because the rate at which the circuit oscillates is determined by the propagation delay in the inverter.



The propagation delay of the NOT gate is 10 ns in this example.

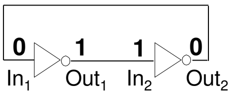
Another **unstable** circuit:



Feedback Loop with Two Inverters (Bistable Circuit)

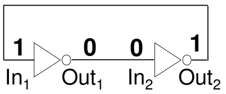
Next, consider a feedback loop with two inverters, shown below.
In this case, the circuit has two stable conditions (bistable), often referred to as stable states.

Stable state 1:



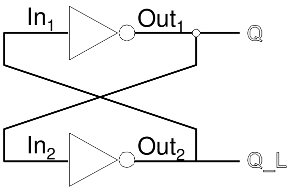
If the input to the first inverter is 0, its output will be 1.
Then, the input to the second inverter will be 1, and its output will be 0.
This 0 will feed back into the first inverter, but since this input is already 0, no changes will occur.
The circuit is then in a stable state.

Stable state 2:



A second stable state of the circuit occurs when the input to the first inverter is 1 and the input to the second inverter is 0.

Bistable (Two stable states) Circuit (cont'd)



We can also draw the bistable circuit in slide 8.12, as shown on the right (cross-coupled form).

Remember, this circuit will always be in one of the **two** possible **stable** states.

State 1: In$_1$ = 0, Out$_1$ = In$_2$ = 1, Out$_2$ = 0

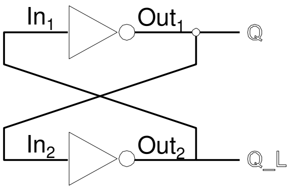
If the output of the inverter at the top is Out$_1$ (Q) = 1, the input of the inverter at the bottom becomes In$_2$ = 1, and its output becomes Out$_2$ (Q_L) = 0.
This is a stable state as it requires In = 0 and Q = 1.

State 2: In$_1$ = 1, Out$_1$ = In$_2$ = 0, Out$_2$ = 1

If the output of the inverter at the top is Out$_1$ (Q) = 0, the input of the inverter at the bottom becomes In$_2$ = 0, and its output becomes Out$_2$ (Q_L) = 1.
This is also a stable state as this state requires In$_1$ = 1 and Q = 0.

This circuit has **two stable states**: Q = 1 and Q = 0,
Q_L is the complement of Q ($Q_L = \overline{Q}$).

Bistable (Two stable states) Circuit (cont'd)



This circuit has two stable states: Q = 1 and Q = 0.

Having two stable states is a necessary property for a memory unit.

However, this circuit has no external inputs.

It is impossible to control (change) the state of the circuit as it has no inputs.

When this circuit is turned on, the latch assumes a random state depending on the time delay of the gates.

Therefore, it cannot be used as a memory unit.

A memory unit must have

1. two stable states,
2. control input(s), which can be used to change or preserve the state of the unit.

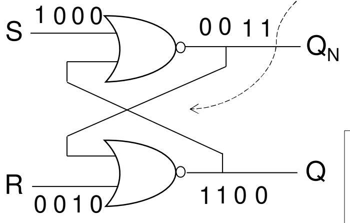
S-R (Set-Reset) Latch

The S-R latch is a one-bit memory device constructed by introducing feedback into a circuit with two NOR (or NAND) gates.

All other latches and flip-flops can be built from this fundamental memory device by adding external gates.

S-R latch with NOR gates:

To emphasize the symmetry between the operation of the two gates, the circuit is often drawn in "cross-coupled" form.



S: Set
R: Reset
Q: Output (State)
Q_N: Complemented output Q_{NOT} ($\bar{Q}$)

Recall: For a NOR gate,
- When one of the inputs is "1", the output will be "0" regardless of the other input.
- The output is "1" only when both inputs are "0".

S	R	Q	Q_N	
1	0	1	0	
0	0	1	0	After S=1, R=0
0	1	0	1	
0	0	0	1	After S=0, R=1
1	1	0	0	Forbidden input

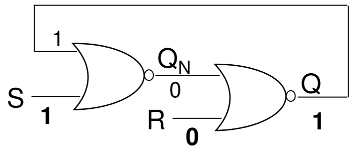
Q and Q_N should be complements!

- The input S is used to write (store) a "1" to the latch (an input S = 1 "sets" the output to Q = 1).
- The input R is used to write a "0" (an input R = 1 "resets" the output to Q = 0).
- If both inputs are "0", the S-R latch preserves its state.
- Both inputs should not be "1" at the same time.

State Changes of the S-R (Set-Reset) Latch

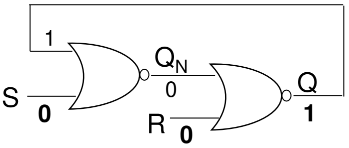
To show how states of an S-R latch change, we can draw the circuits:

S=1, R=0 / Q=1:



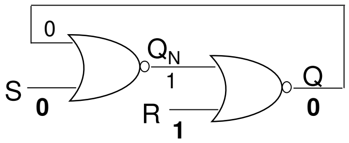
If S = 1, R = 0, Q_N will be 0. Since both inputs of the second gate are 0, Q will become 1.

S=0, R=0 / Q=1:



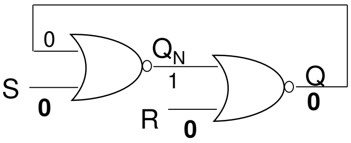
If we change S to 0, the latch will remain in the present state (1) because Q = 1 feeds back into the first gate, causing Q_N to remain 0, as shown above.

S=0, R=1 / Q=0 :



If we now change R to 1, Q will become 0 and Q_N will then change back to 1.

S=0, R=0 / Q=0



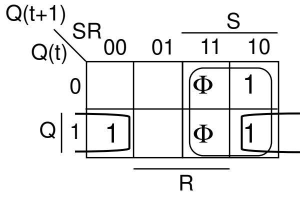
If we then change R back to 0, the latch remains in the present state (0).

Truth table and the characteristic equation of the S-R latch:

The next value of the output (next state) Q(t+1) depends on the inputs and current output (state) Q(t).

Truth table:

Q(t)	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Φ (forbidden)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Φ (forbidden)



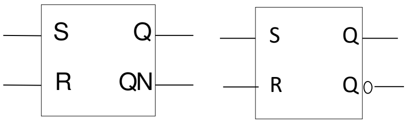
Q(t) \ SR	00	01	11	10
0			Φ	1
1	1		Φ	1

Characteristic Equation:

$$\boxed{Q(t+1)=S+Q(t)\bar{R} \quad (SR=0)}$$

The condition SR = 0 implies that S and R cannot both be 1 at the same time.

Block Diagrams for the S-R Latch:



Timing diagram (behavior) of the S-R latch:

Characteristic Equation:

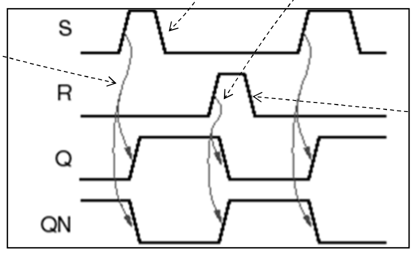
$$\boxed{Q(t+1)=S+Q(t)\bar{R} \quad (SR=0)}$$

When S changes to 1, Q changes to 1 a short time later.

When S changes back to 0, Q does not change.

R changes to 1, and Q changes back to 0 a short time later.

When R changes back to 0, Q does not change.



Timing Diagram

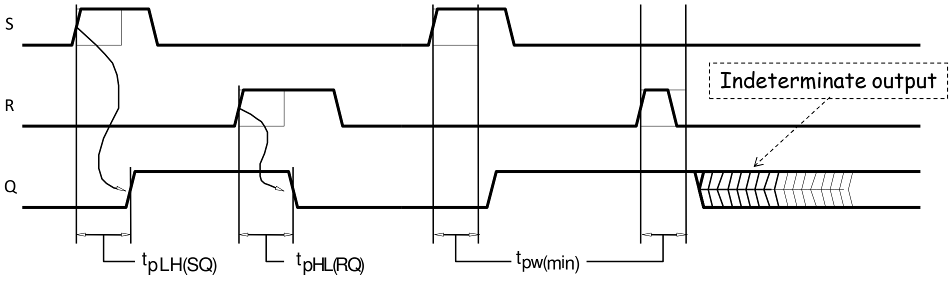
Timing Properties of the S-R latch:

Because of the propagation delay, the changes in the S and R inputs will not affect the output instantaneously.

During the delay, the inputs should be kept constant.

The duration of the S (or R) input pulse must normally be at least as great as the delay for a change in the state of Q to occur.

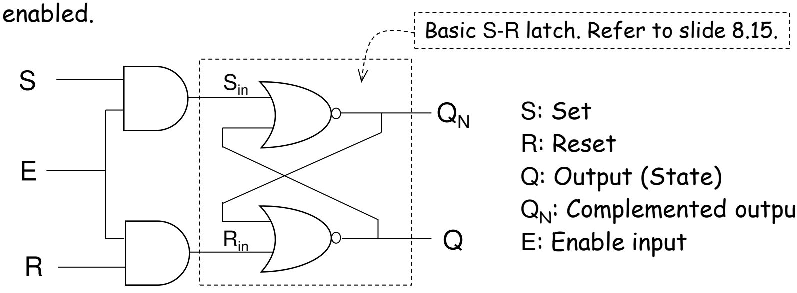
Otherwise, the value of the output is indeterminate (random).



$t_{pLH(SQ)}$: Delay in the 0-1 transition of the output when S changes.
$t_{pHL(RQ)}$: Delay in the 1-0 transition of the output when R changes.
$t_{pW(min)}$: Minimum duration for which the inputs should be kept constant.

S-R Latch with Enable Input

AND gates are connected to S and R inputs so that the latch is active only when it is enabled.



Basic S-R latch. Refer to slide 8.15.

S: Set
R: Reset
Q: Output (State)
Q_N: Complemented output (Q')
E: Enable input

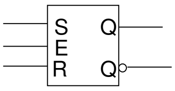
The value of the latch can only be changed when E=1.

When E=1: S_{in} = S and R_{in} = R (normal operation of the basic S-R latch)

When E=0, the value of the latch is preserved regardless of the inputs.

When E=0: S_{in} = 0 and R_{in} = 0 (the basic S-R latch preserves its value)

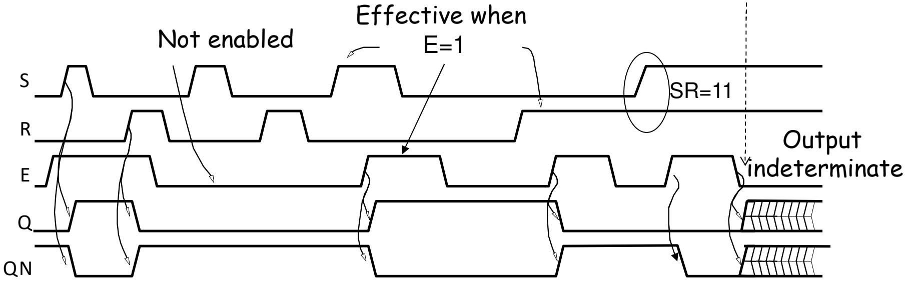
Timing Diagram of the S-R Latch with Enable Input



E	S	R	Q(t+1)
0	X	X	Q(t)
1	1	0	1
1	0	1	0
1	0	0	Q(t)
1	1	1	forbidden

If the forbidden input (SR=11) is applied to the latch, both Q and Q' outputs will be 1.
If the latch is disabled in this state, the value in the latch is indeterminate.

Timing Diagram:


Not enabled

Effective when E=1

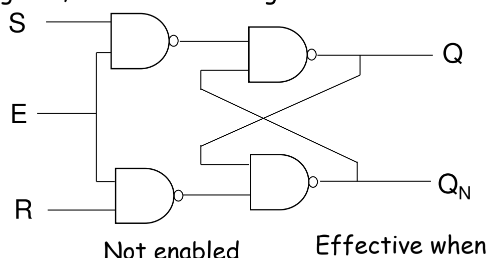
SR=11

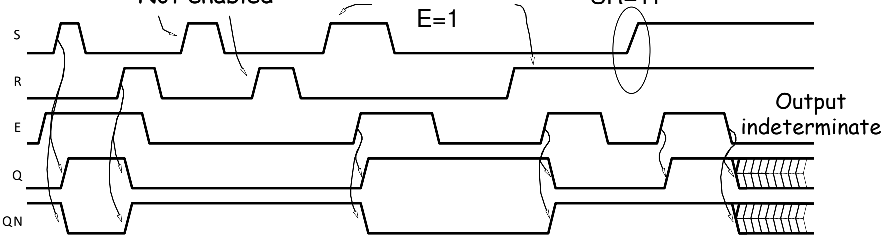
Output indeterminate

S-R Latch with Enable Input (with only NAND Gates)

The S-R latch implemented with NOR and AND gates (slide 8.20) can also be implemented using only NAND gates.

Remember: If we add NOT gates to the outputs of AND gates and to the inputs of the OR gates, we obtain NAND gates.




Not enabled

Effective when E=1

SR=11

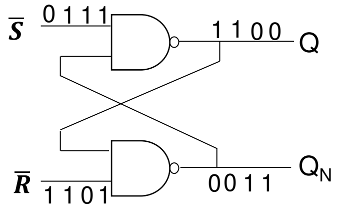
Output indeterminate

$\bar{\mathbf{S}} - \bar{\mathbf{R}}$ Latch

Remember: On slide 8.15, we implemented an S-R latch using NOR gates.

An S-R latch can be also implemented using NAND gates instead of NOR.

This results in an $\bar{\mathbf{S}} - \bar{\mathbf{R}}$ latch.



$\bar{S}$: Complement of Set
$\bar{R}$: Complement of Reset
Q: Output (State)
Q_N: Complemented output ($\bar{Q}$)

This latch is different from the latches on slides 8.20 and 8.22.

$\bar{S}$	$\bar{R}$	Q	Q_N	
0	1	1	0	
1	1	1	0	After $\bar{S}$=0, $\bar{R}$=1
1	0	0	1	
1	1	0	1	After $\bar{S}$=1, $\bar{R}$=0
0	0	1	1	Forbidden inputs

Q and Q_N should be complements!

Recall: For a NAND gate,

- If one of the inputs of a NAND gate is "0", the output will be "1" regardless of the other input.
- The output is "0" only when both inputs are "1".

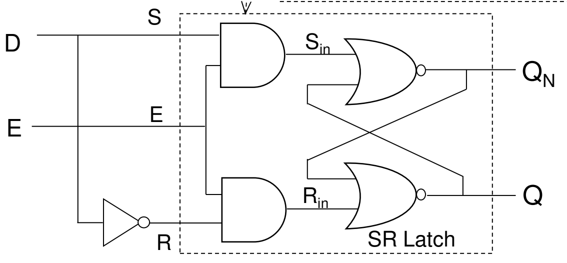
Delay (D) Latch

Other types of latches can be built from the S-R latch by adding external gates.

One way to eliminate the undesirable condition of the indeterminate state in the S-R latch is to ensure that inputs S and R are never equal to 1 at the same time.

If an inverter is added to an S-R latch to make R the complement of S, the D-latch is obtained.

S-R latch with enable input
Refer to slide 8.20.



E=1	D	S_{in}	R_{in}	Q(t+1)
	0	0	1	0 (Reset)
	1	1	0	1 (Set)

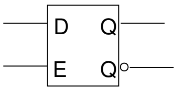
E=0	D	S_{in}	R_{in}	R Q(t+1)
	X	0	0	Q(t) (Don't change)

If E=1, the value of input D is written to the latch.

If E=0, the latch preserves its previous value.

Characteristic Equation: $\boxed{Q(t+1) = D}$

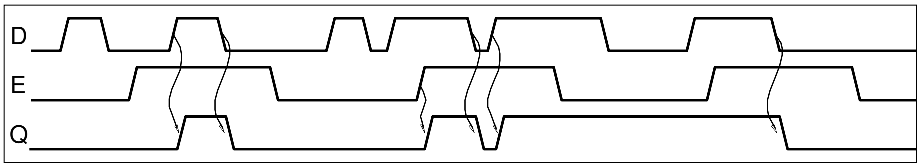
Timing diagram of the Delay (D) Latch



E	D	$Q(t^+)$	$Q_N(t^+)$
1	0	0	1
1	1	1	0
0	X	Q(t)	$Q_N(t)$

Characteristic Equation:

$$\boxed{Q(t+1) = D}$$



A D-latch without an enable input is not preferable because it can not preserve its value if its input changes.

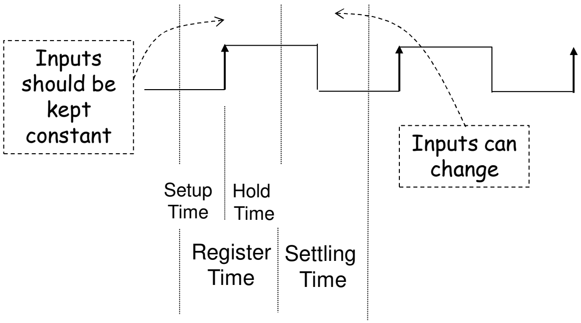
Difference between a Latch and a Flip-Flop:

- **A latch** is not triggered by a clock signal. The value of the latch can be changed whenever the latch is enabled.
- **A flip-flop** is a memory unit that is triggered by a clock signal.

Remember that the **clock signal** is a periodic square wave that synchronizes the gates in the circuit.

Edge-triggered units:

These units use an edge (rising edge in positive logic, or falling edge in negative logic) of the clock signal as active.



Inputs should be kept constant

Inputs can change

Setup Time

Hold Time

Register Time

Settling Time

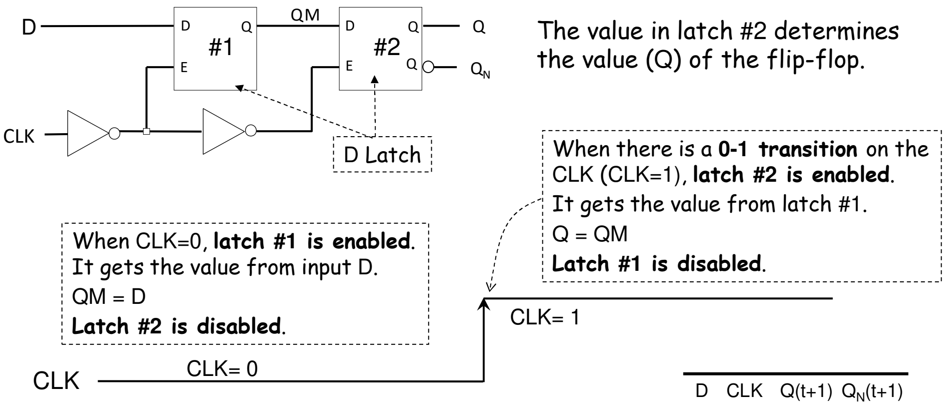
The **register time** is the sum of the setup and hold times.

Setup time is the minimum time the data signal should be steady **before** the clock transition.

Hold time is the minimum time the data signal should be steady **after** the clock transition.

Rising-Edge Triggered D Flip-Flop

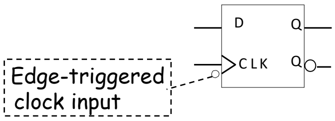
Although the edge-triggered flip-flop consists of two latches, it is a 1-bit memory.



The value in latch #2 determines the value (Q) of the flip-flop.

When there is a **0-1 transition** on the CLK (CLK=1), **latch #2 is enabled**.
It gets the value from latch #1.
Q = QM
Latch #1 is disabled.

When CLK=0, **latch #1 is enabled**.
It gets the value from input D.
QM = D
Latch #2 is disabled.



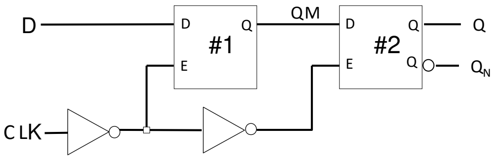
Edge-triggered clock input

Characteristic Equation:

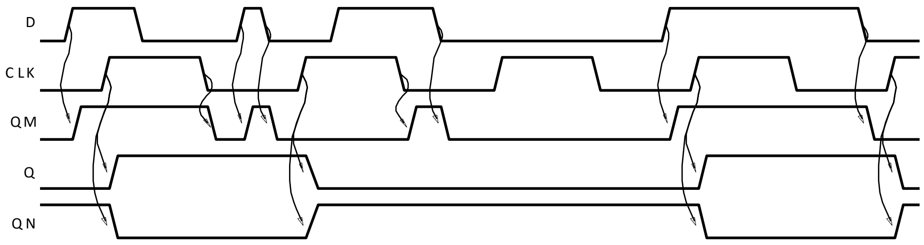
$$\boxed{Q(t+1) = D}$$

D	CLK	Q(t+1)	$Q_N(t+1)$
0	↑	0	1
1	↑	1	0
x	0	Q(t)	$Q_N(t)$
x	1	Q(t)	$Q_N(t)$

Timing diagram for the rising-edge triggered D flip-flop



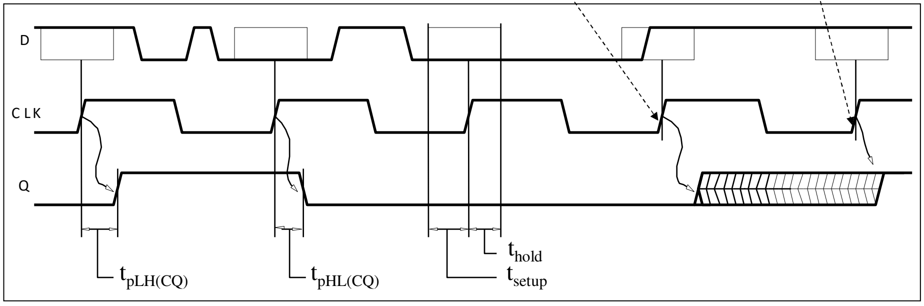
- When CLK=0, latch #1 is enabled. It gets the value from input D. QM = D
- When CLK=1, latch #2 is enabled. It gets the value from latch #1. Q = QM



Timing properties of the rising-edge triggered D flip-flop

The output is indeterminate as the input is not kept constant during the *setup time.*

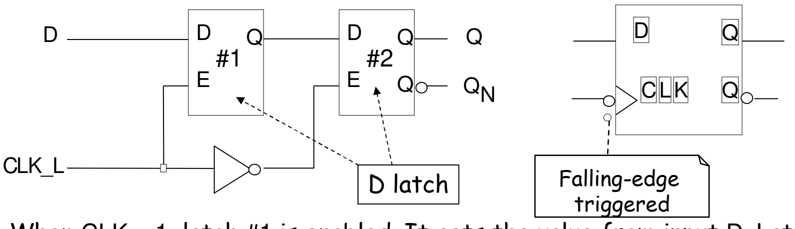
Valid data is produced at the output ("1", in this example).



$t_{pLH(CQ)}$: Delay between the active edge of clock and the 0-1 transition of the output.
$t_{pHL(CQ)}$: Delay between the active edge of clock and the 1-0 transition of the output.
t_{setup} : Time before the active edge when inputs should be kept constant.
t_{hold} : Time after the active edge when inputs should be kept constant.

Falling-Edge Triggered D Flip-Flop

Data is written to the D flip-flop during the falling edge of the clock signal.



D latch

Falling-edge triggered

- When CLK = 1, latch #1 is enabled. It gets the value from input D. Latch #2 is disabled.
- When CLK goes to 0, latch #2 is enabled. It gets the value from latch #1. The value in latch #2 determines the value of the flip-flop.

D	CLK_L	Q(t+1)	Q_N(t+1)
0	↓	0	1
1	↓	1	0
x	0	Q(t)	Q_N(t)
x	1	Q(t)	Q_N(t)

Characteristic Equation:

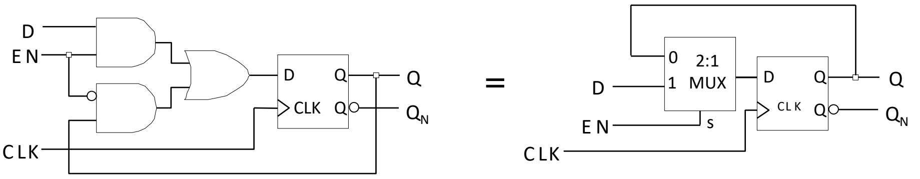
$$\boxed{Q(t+1) = D}$$

Edge-Triggered D Flip-Flop with Enable Input

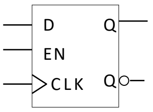
A flip-flop may also have an enable input (EN).

If the enable input (EN) is active, the value of the flip-flop can be changed.

Otherwise, the flip-flop preserves its value regardless of its inputs.

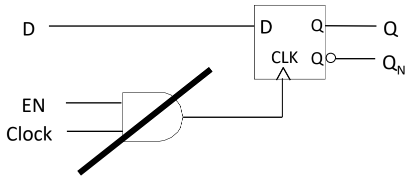


D	EN	CLK	Q(t+1)	Q_N(t+1)
0	1	↑	0	1
1	1	↑	1	0
x	0	↑	Q(t)	Q_N(t)
x	x	0	Q(t)	Q_N(t)
x	x	1	Q(t)	Q_N(t)



Edge-Triggered D Flip-Flop with Enable Input (cont'd)

Can we design a simpler circuit by connecting an AND gate to the CLK input of the D flip-flop?



This is **not a proper solution**.

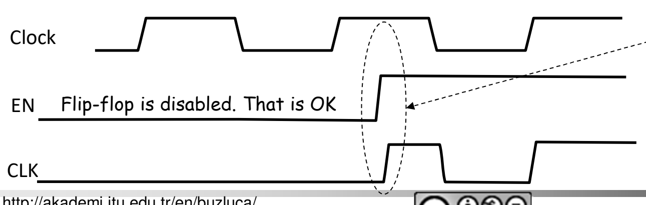
This circuit has some problems.

1. It is not a good idea to increase the delay at the CLK input.
   It can cause synchronization problems.

2. If Clock=1 and EN=0, then CLK=0 and the flip-flop is disabled (that is fine).

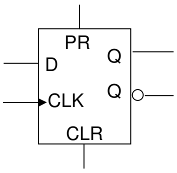
On the other hand, if EN becomes 1 while Clock=1, the flip-flop will get a rising edge on the CLK input, and it will process its input D.

However, the real clock signal is 1; there is no 0-1 transition on the clock input.



Problem:
There is no real edge, but the FF gets a rising edge.

Asynchronous "Clear" and "Preset" inputs



Clocked integrated circuit flip-flops often have additional inputs, which can be used to set the flip-flop to an initial state **independent of the clock** (asynchronous).

- PR (*Preset*) is used to write a “1” to the flip-flop.
  A "1" applied to the preset (PR) input will set the flip-flop to Q = 1.
- CLR (*Clear*) is used to write a “0”.
  A "1" applied to the clear (CLR) input will reset the flip-flop to Q = 0.

These asynchronous inputs override the clock and D inputs.

That is, a "1" applied to the clear input will reset the flip-flop regardless of the values of D and the clock.

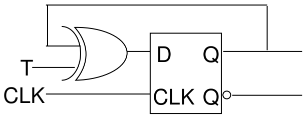
These asynchronous inputs can be used to assign initial values to the flip-flops after power-on reset.

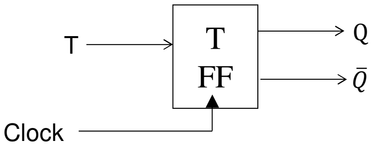
Some devices use negative logic for PR and CLR inputs.

For example, in such a device, a "0" must be applied to the clear (CLR) input to reset the flip-flop to Q = 0.

Edge-Triggered Toggle (T) Flip-Flop

An edge-triggered T flip-flop can be implemented using an edge-triggered D flip-flop and an XOR gate.





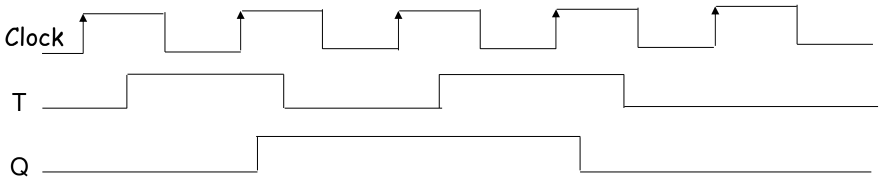
If the input is 0 (T=0), the value of the flip-flop is preserved as $0 \oplus Q = Q$.

$$T = 0 \rightarrow Q(t+1) = Q(t)$$

If the input is 1 (T=1), the value of the flip-flop is complemented (toggled) because $1 \oplus Q = \overline{Q}$.

$$T = 1 \rightarrow Q(t+1) = \overline{Q(t)}$$

Characteristic Equation: $\boxed{Q(t+1)= T \oplus Q(t)}$

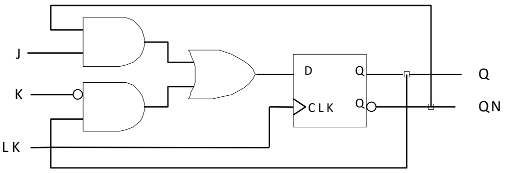


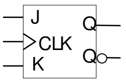
Edge-Triggered J-K Flip-Flop

The J-K flip-flop combines the features of the S-R and T flip-flops.

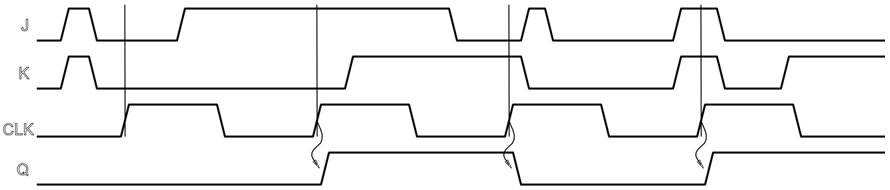
- A "1" input applied to J or K alone acts exactly like an S or R input, respectively.
  - if **J = 1**, K = 0, the latch output is set to Q = 1;
  - if J = 0, **K = 1**, the latch output is reset to Q = 0.
- If a "1" input is applied to both simultaneously (**J=1** and **K=1**), the flip-flop changes state (toggles) just like a T flip-flop.

Characteristic Equation: $\boxed{Q(t+1) = J \bullet \overline{Q(t)} + \overline{K} \bullet Q(t)}$





J	K	CLK	Q(t+1)	QN(t+1)
x	x	0	Q(t)	QN(t)
x	x	1	Q(t)	QN(t)
0	0	↑	Q(t)	QN(t)
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	QN(t)	Q(t)



Characteristic Equations of Latches and Flip-Flops:

The functional behavior of a latch or flip-flop can be described by a **characteristic equation** that specifies the next state of the flip-flop (or latch) as a function of its inputs and current state.

Characteristic equations of flip-flops:

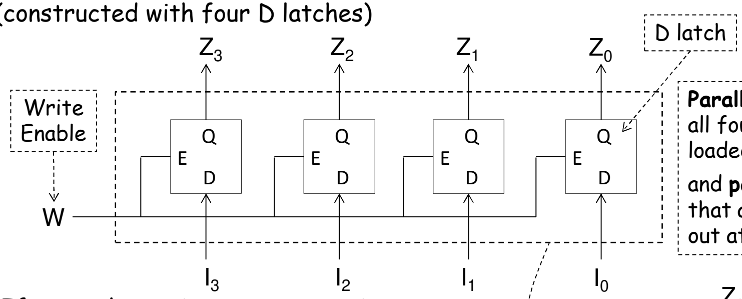
S-R FF :	$Q(t+1) = S + \overline{R} \bullet Q(t)$ (SR = 0)
J-K FF :	$Q(t+1) = J \bullet \overline{Q(t)} + \overline{K} \bullet Q(t)$
D FF :	$Q(t+1) = D$
T FF :	$Q(t+1) = T \oplus Q(t)$

Registers

A register is a group of binary memory units that can store binary information.

An *n*-bit register has a group of *n* flip-flops (or latches) and can store *n* bits.

Example: 4-bit parallel in/out register with write enable (constructed with four D latches)



Parallel-in implies that all four bits can be loaded at the same time,

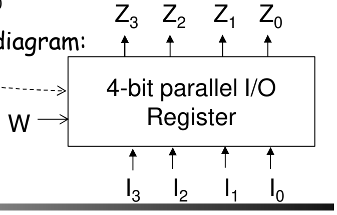
and **parallel-out** implies that all bits can be read out at the same time.

If W=0, the register preserves its current state (contents).

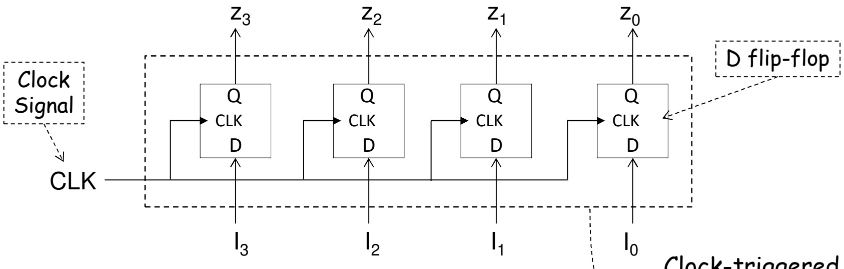
If W=1, the data applied to the inputs will be written to the register.

W = 0, Z(t+1) = Z(t)
W = 1, Z(t+1) = I(t)

Block diagram:

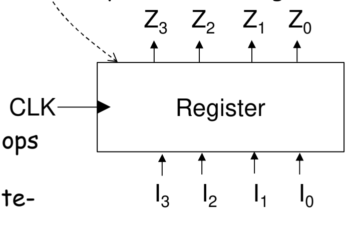


Example: 4-bit parallel in/out register **with clock signal** (constructed with four D flip-flops)



If the clock signal is active (for example, rising edge), the data is written to the register.

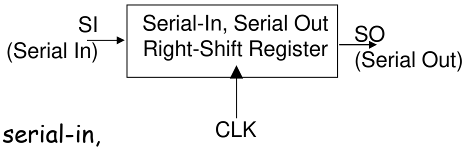
Clock-triggered 4-bit parallel I/O register:



Note: If we construct the register using D flip-flops with enable (E) inputs, then the register will have inputs for both the clock signal (CLK) and the write-enable (W) signal.

Shift Registers: Serial vs. Parallel Inputs/Outputs

Shift registers can store and shift binary numbers to the left or right by one bit at each active transition of the clock signal.



Serial-in, Serial-out:

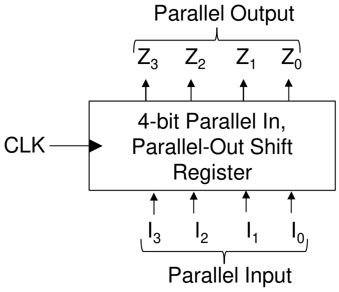
The figure on the right illustrates an n-bit serial-in, serial-out right-shift register.

- Serial in means that data is shifted into the first flip-flop one bit at a time, and the flip-flops cannot be loaded in parallel.
- Serial out means that data can only be read out of the last flip-flop and the outputs from the other flip-flops are not available.

Parallel-in, parallel-out:

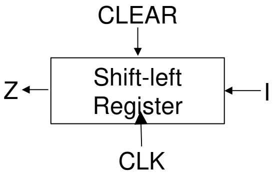
The figure on the right illustrates a 4-bit parallel-in, parallel-out shift register.

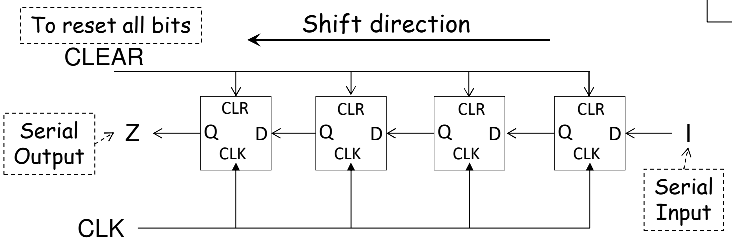
- If all the bits of the register are loaded simultaneously with a single clock pulse, we say that the loading is done in parallel.
- Parallel-in implies that all four bits can be loaded at the same time, and parallel-out implies that all bits can be read out at the same time.



Shift Registers (cont'd)

Example: 4-bit serial-in, serial-out left-shift register with CLEAR input





The data is shifted to the left by one bit at each active transition of the clock signal.

In this design, it is not possible to load initial values (except zero) in parallel into the flip-flops (serial-in).

Data can only be read from the last flip-flop, and the outputs from the other flip-flops are not available (**serial-out**).

A right-shift register can be implemented by changing the order of the flip-flops.

Shift Register Configurations

Four register configurations (the combinations resulting from a choice of parallel or serial for the inputs and the outputs) are possible:

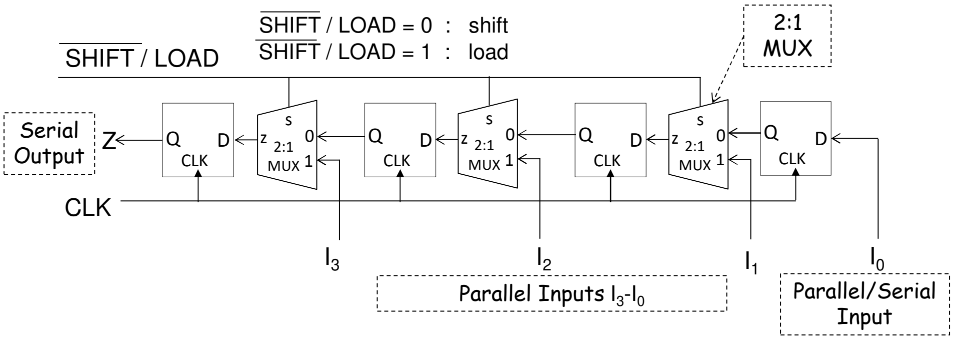
- Parallel-in, Parallel-out
- Serial-in, Serial-out
- Parallel-in, Serial-out
- Serial-in, Parallel-out

Shift Register Applications

- Many computers operate on parallel data, but this data must sometimes be converted to serial format to be sent.
- Throughout most of the history of personal computers, data has been transferred through serial ports to devices such as modems, terminals, and various peripherals and directly between computers.
- ICs called UARTs (universal asynchronous receiver-transmitters) are used to interface microprocessors and parallel data to communications links that use serial data.
- Shift registers can perform this parallel-to-serial conversion and serial-to-parallel conversion.
- Shift registers are available in IC form or can be constructed from discrete flip-flops.

Example: 4-bit left-shift register with parallel load (inputs), serial output

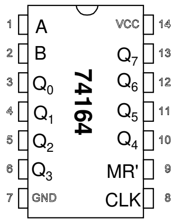
In this design, we can load an initial value to the register in parallel.


$\overline{\text{SHIFT}}$ / LOAD = 0 : shift
$\overline{\text{SHIFT}}$ / LOAD = 1 : load
$\overline{\text{SHIFT}}$ / LOAD
2:1 MUX
Serial Output
Z
Q
D
CLK
s
z
2:1
MUX
0
1
CLK
I_3
I_2
I_1
I_0
Parallel Inputs I_3-I_0
Parallel/Serial Input

The shift register has the control input $\overline{\text{SHIFT}}/\text{LOAD}$, which enables either a shift or a load:

- If $\overline{\text{SHIFT}}/\text{LOAD} = 0$, clocking the register causes the serial input (I_0) to be shifted into the first flip-flop Q_0, while the data in flip-flops Q_3, Q_2, and Q_1 are shifted left.
- If $\overline{\text{SHIFT}}/\text{LOAD} = 1$, clocking the shift register will cause the four data inputs (I_3, I_2, I_1, and I_0) to be loaded in parallel into the flip-flops.

Example: 74164 8-Bit Serial-In Parallel-Out Shift Register



Serial data is entered through a 2-input AND gate (A and B) synchronous with the LOW-to-HIGH transition of the clock (CLK).

This register does not have parallel load capability (Q_7-Q_0 are outputs).

This device also has an asynchronous Master Reset (MR'), which clears the register by resetting all outputs to LOW independent of the clock.

For details, you may refer to the datasheet of the device.