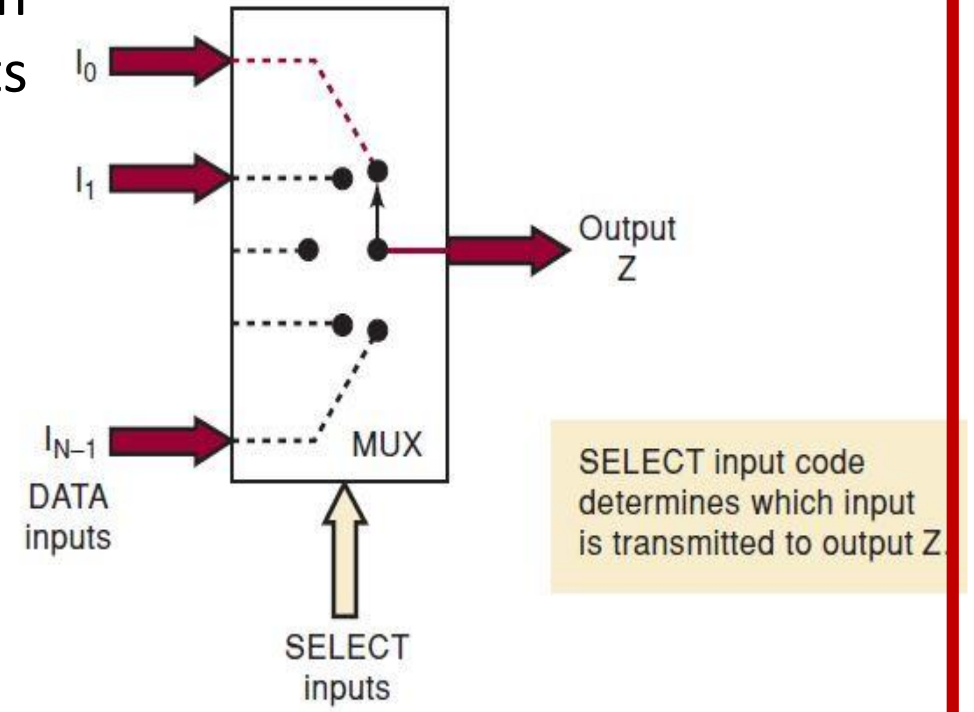
MULTIPLEXERS (DATA SELECTORS)

- A *digital multiplexer* or *data selector* is a logic circuit that accepts several digital data inputs and selects one of them at any given time to pass on to output.
- Routing of desired data input to output is controlled by SELECT inputs (ADDRESS inputs).
- Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.
- Multiplexer acts as a digitally controlled multi-position switch, where digital code applied to SELECT inputs controls data inputs to be switched to output.
- Output Z will equal data input I_0 for some particular SELECT input code, Z will equal I_1 for another particular SELECT input code, and so on.
- Multiplexer selects 1 out of N input data sources and transmits selected data to a single output channel called **multiplexing**.



I_0
I_1
I_{N-1}
DATA inputs
MUX
Output Z
SELECT inputs
SELECT input code determines which input is transmitted to output Z.

Basic *Two- to One* line Multiplexer

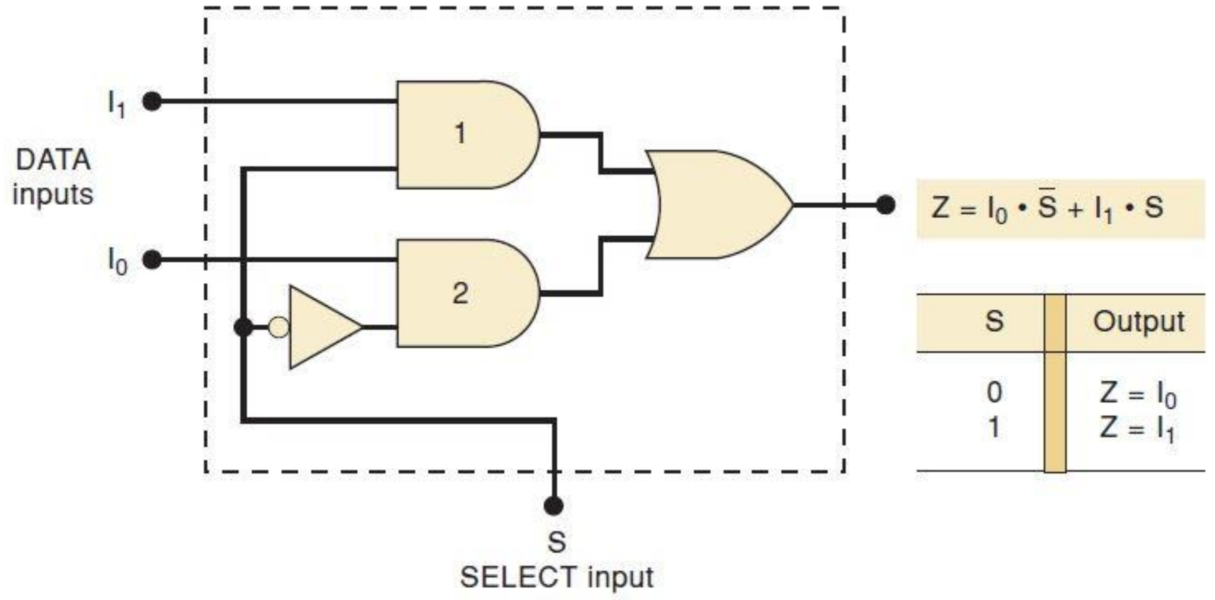
- Logic level applied to *S* input determines which AND gate is enabled so that its data input passes through OR gate to output *Z.*
- Boolean expression for output is

  $Z = I_0S + I_1S$
- With S = 0

  $Z = I_0 \, . \, 1 + I_1 \, . \, 0$

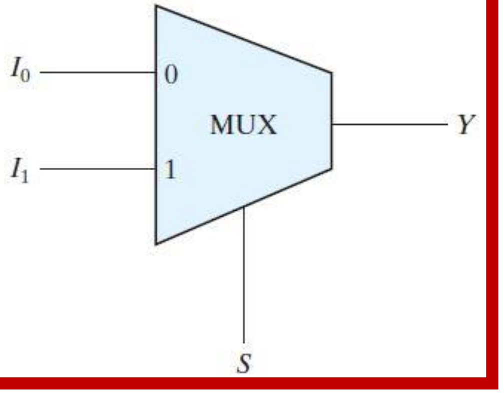
  $= I_0$ [gate 2 enabled]
- Z will be identical to input signal I_0.
- With S = 1

  $Z = I_1$ [gate 1 enabled]
- Output Z will be identical to input signal I_1.
- Multiplexer acts like an electronic switch that selects one of two sources.



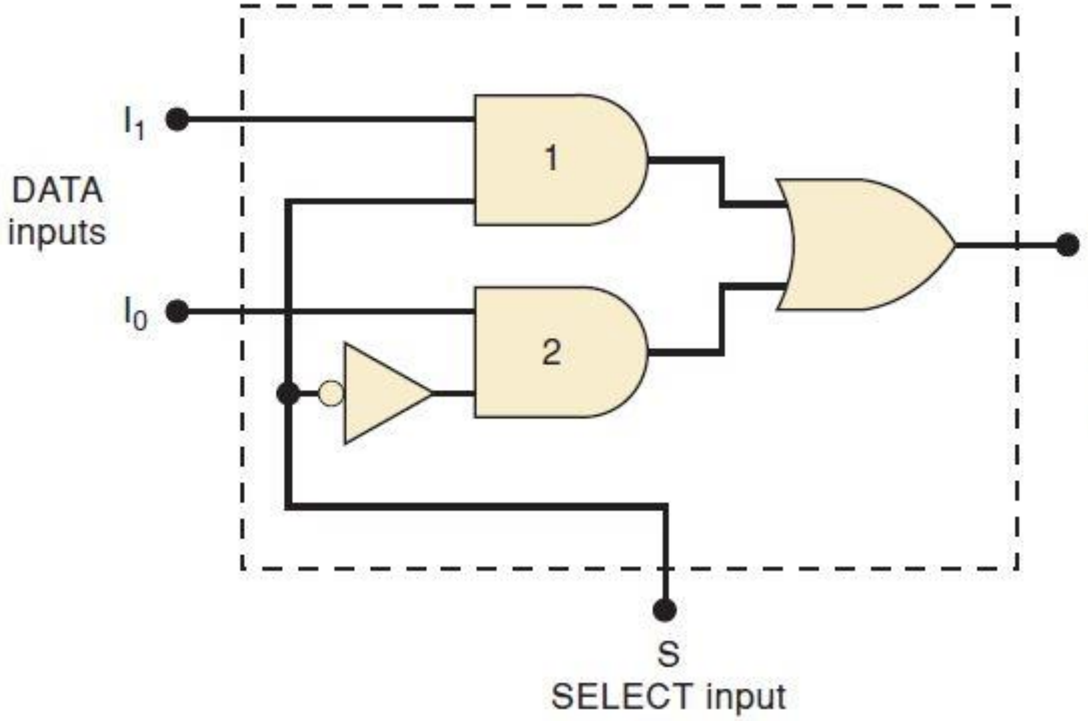
S	Output
0	$Z = I_0$
1	$Z = I_1$

Logic circuit for two-input multiplexer with data inputs I_0 and I_1 and SELECT input S.



Basic *Two- to One* line Multiplexer

- Ex. a digital system that uses two different MASTER CLOCK signals: a high-speed clock (say, 10 MHz) in one mode and a slow-speed clock (say, 4.77 MHz) for the other.
- 10-MHz clock would be tied to I_0 and 4.77-MHz clock would be tied to I_1.
- A signal from the system's control logic section would drive SELECT input to control which clock signal appears at output *Z* for routing to other parts of the circuit.

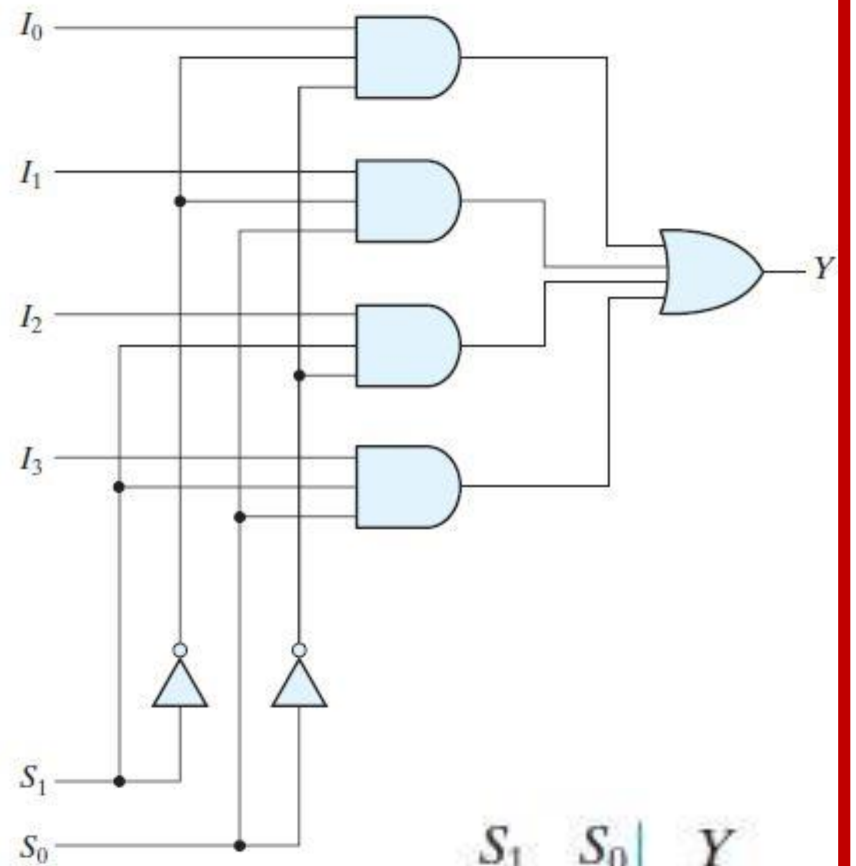


$$Z = I_0 \cdot \overline{S} + I_1 \cdot S$$

S	Output
0	$Z = I_0$
1	$Z = I_1$

Four-to One line Multiplexer

- Each of four inputs, I_0 through I_3, is applied to one input of AND gate.
- Selection lines S_1 and S_0 are decoded to select a particular AND gate.
- Outputs of AND gates are applied to single OR gate that provides one-line output.
- Consider when $S_1S_0 = 10$.
- AND gate associated with input I_2 has two of its inputs equal to 1 and third input connected to I_2.
- Other three AND gates have at least one input equal to 0, which makes their outputs equal to 0.
- Output of OR gate is now equal to value of I_2, providing a path from selected input to output.



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Four-to One line Multiplexer

- AND gates and inverters in multiplexer resemble a decoder circuit, and indeed, they decode selection input lines.

- 2^n -to-1-line multiplexer is constructed from an n -to-2^n decoder by adding 2^n input lines to it, one to each AND gate.

- Outputs of AND gates are applied to a single OR gate.

- Size of a multiplexer is specified by number 2^n of its data input lines and single output line.

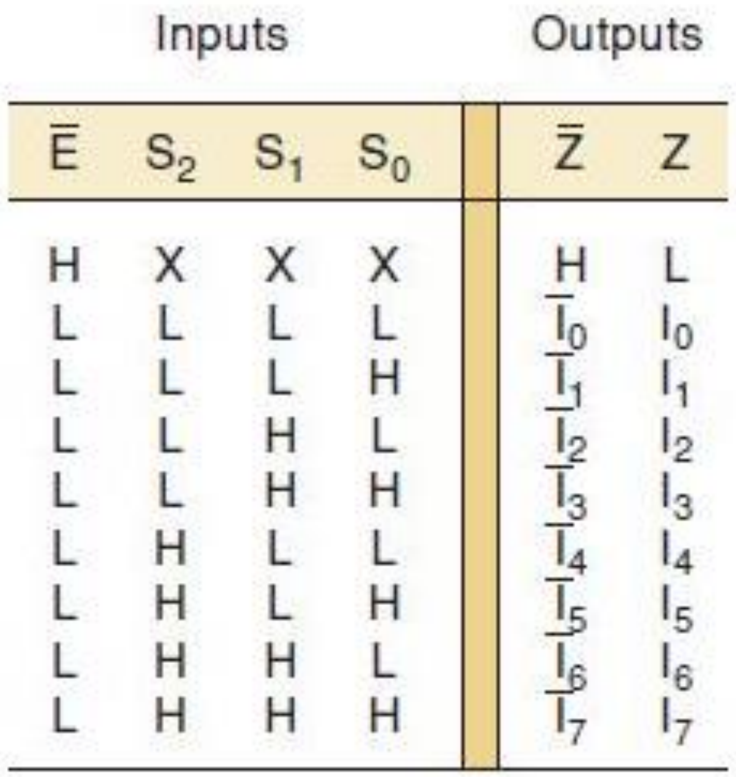
- n- selection lines are implied from 2^n data lines.

- As in decoders, multiplexers may have an enable input to control operation of the unit.

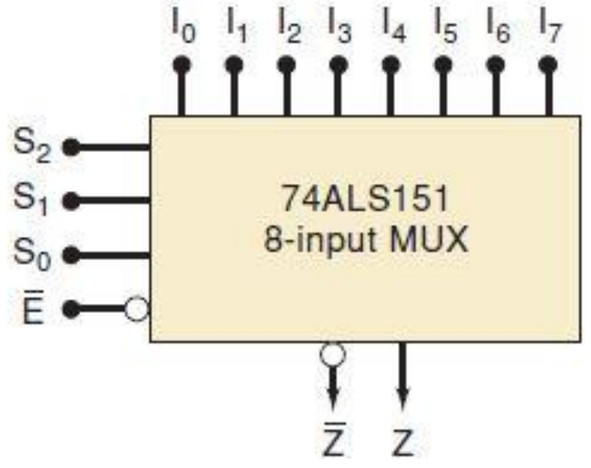
- When enable input is in inactive state, outputs are disabled, and when it is in the active state, circuit functions as a normal multiplexer.

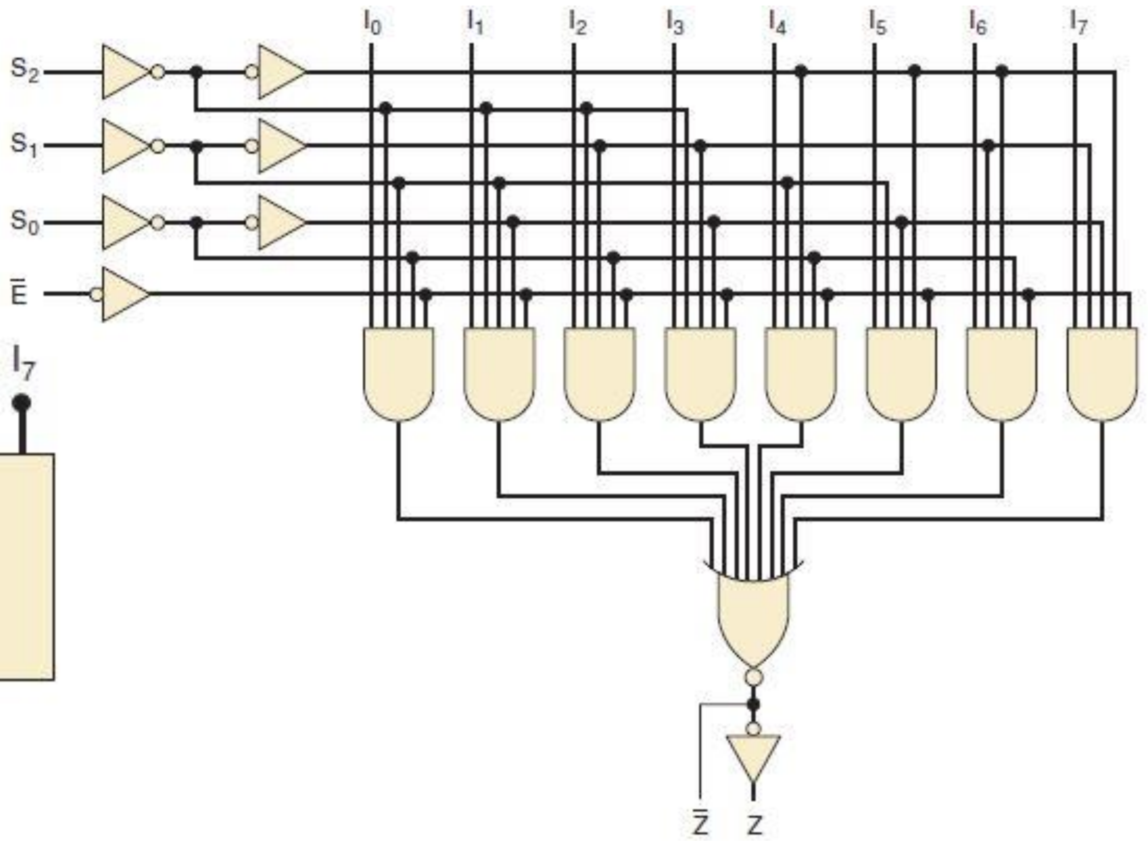
Eight- to one line Multiplexer

- This multiplexer has an enable input *E′* and provides both normal and inverted outputs.
- When *E′* = 0, select inputs $S_2S_1S_0$ will select one data input (from I_0 through I_7) for passage to output *Z.*
- When E′ = 1, multiplexer is disabled so that Z = 0, regardless of select input code.



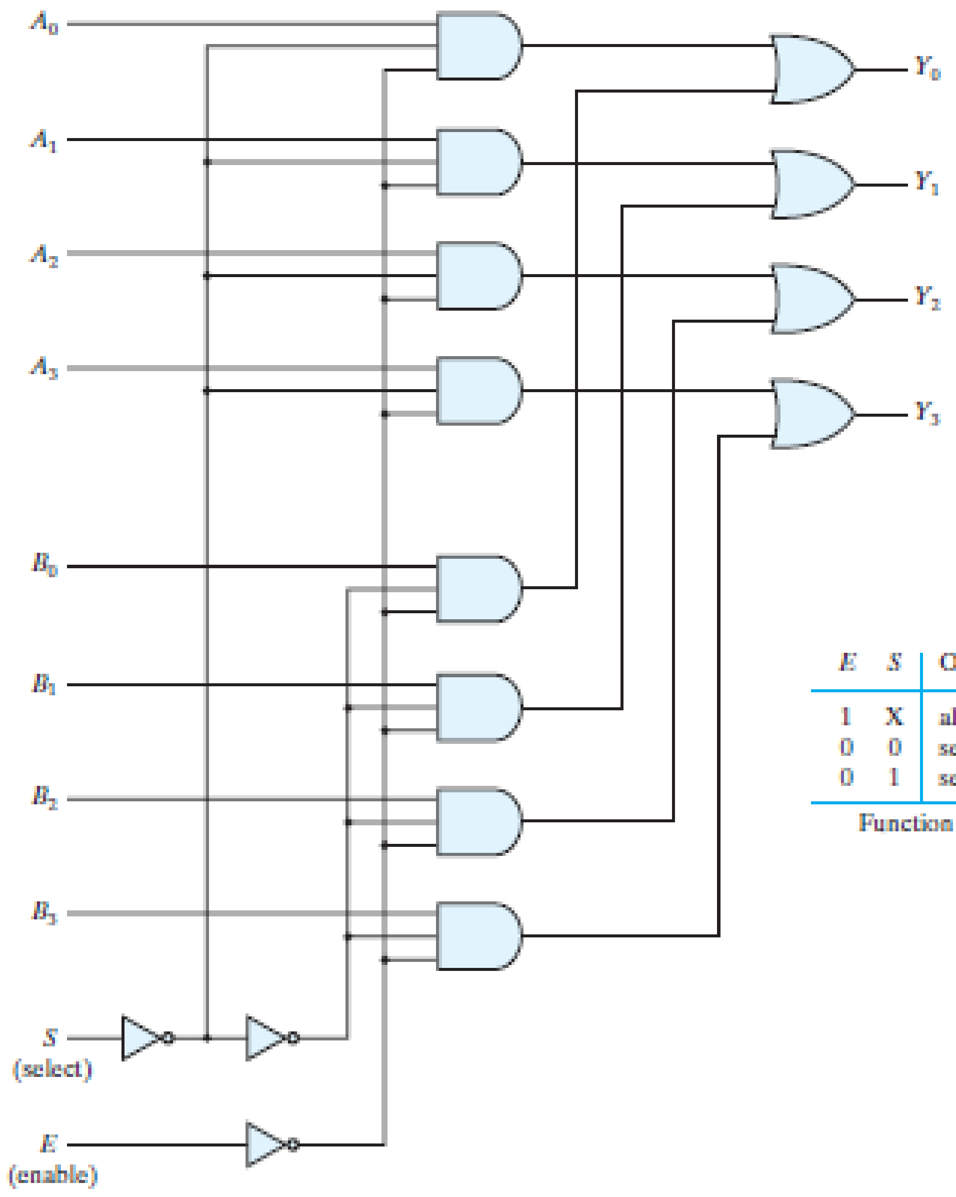
Inputs				Outputs	
$\overline{E}$	S_2	S_1	S_0	$\overline{Z}$	Z
H	X	X	X	H	L
L	L	L	L	$\overline{I_0}$	I_0
L	L	L	H	$\overline{I_1}$	I_1
L	L	H	L	$\overline{I_2}$	I_2
L	L	H	H	$\overline{I_3}$	I_3
L	H	L	L	$\overline{I_4}$	I_4
L	H	L	H	$\overline{I_5}$	I_5
L	H	H	L	$\overline{I_6}$	I_6
L	H	H	H	$\overline{I_7}$	I_7





Quadruple *Two*-to-one-line multiplexer

- Multiplexer circuits can be combined with common selection inputs to provide multiple-bit selection logic.
- The circuit has four multiplexers, each capable of selecting one of two input lines.
- Output Y_0 can be selected to come from either input A_0 or input B_0.
- Similarly, output Y_1 may have the value of A_1 or B_1, and so on.
- Input selection line S selects one of the lines in each of the four multiplexers.
- Enable input E must be active for normal operation.
- Although circuit contains four 2-to-1-line multiplexers, it is viewed as a circuit that selects one of two 4-bit sets of data lines.

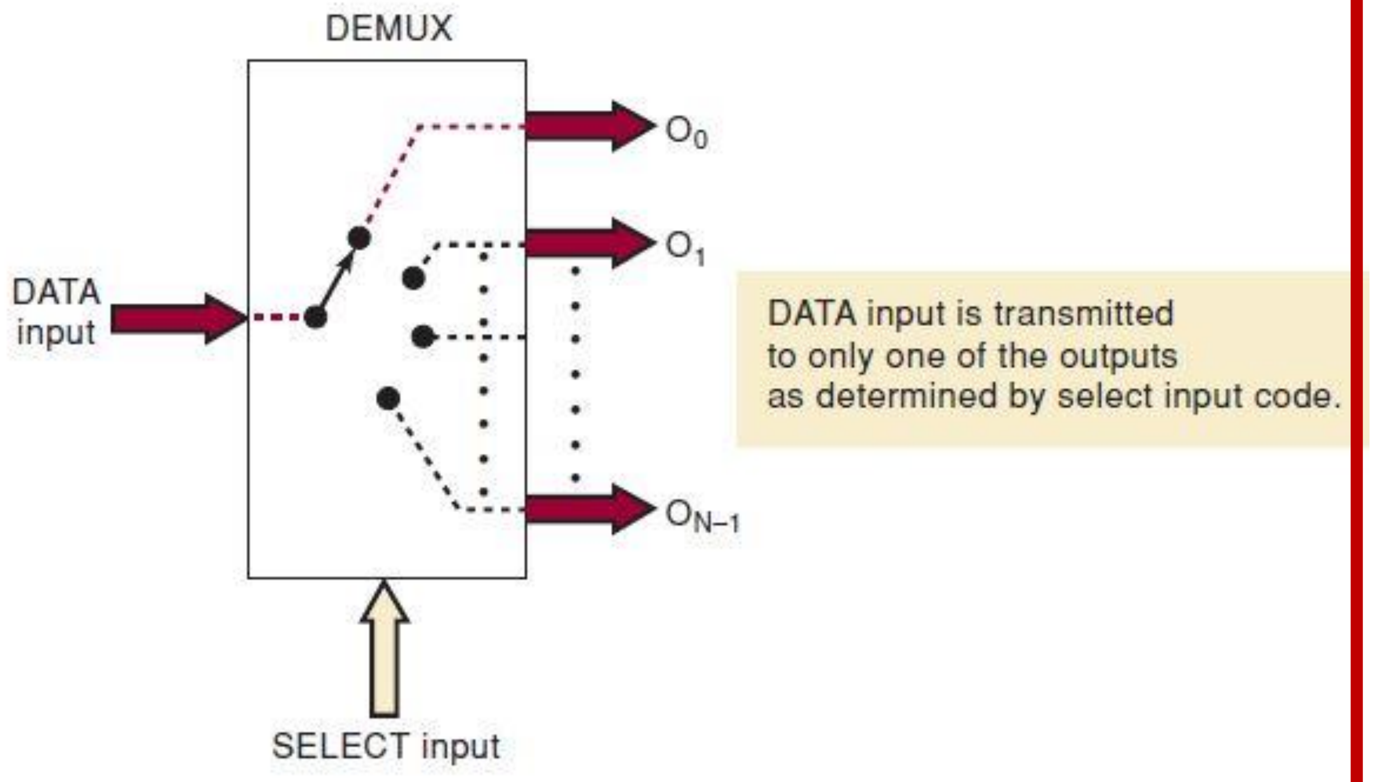


E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

Function table

DEMULTIPLEXERS (DATA DISTRIBUTORS)

- A multiplexer takes several inputs and transmits *one* of them to the output.
- *A* **demultiplexer (DEMUX)** performs reverse operation: it takes a single input and distributes it over several outputs.
- Large arrows for inputs and outputs represent one or more lines.
- The select input code determines to which output DATA input will be transmitted.
- Demultiplexer takes one input data source and selectively distributes it to 1 of *N* output channels.

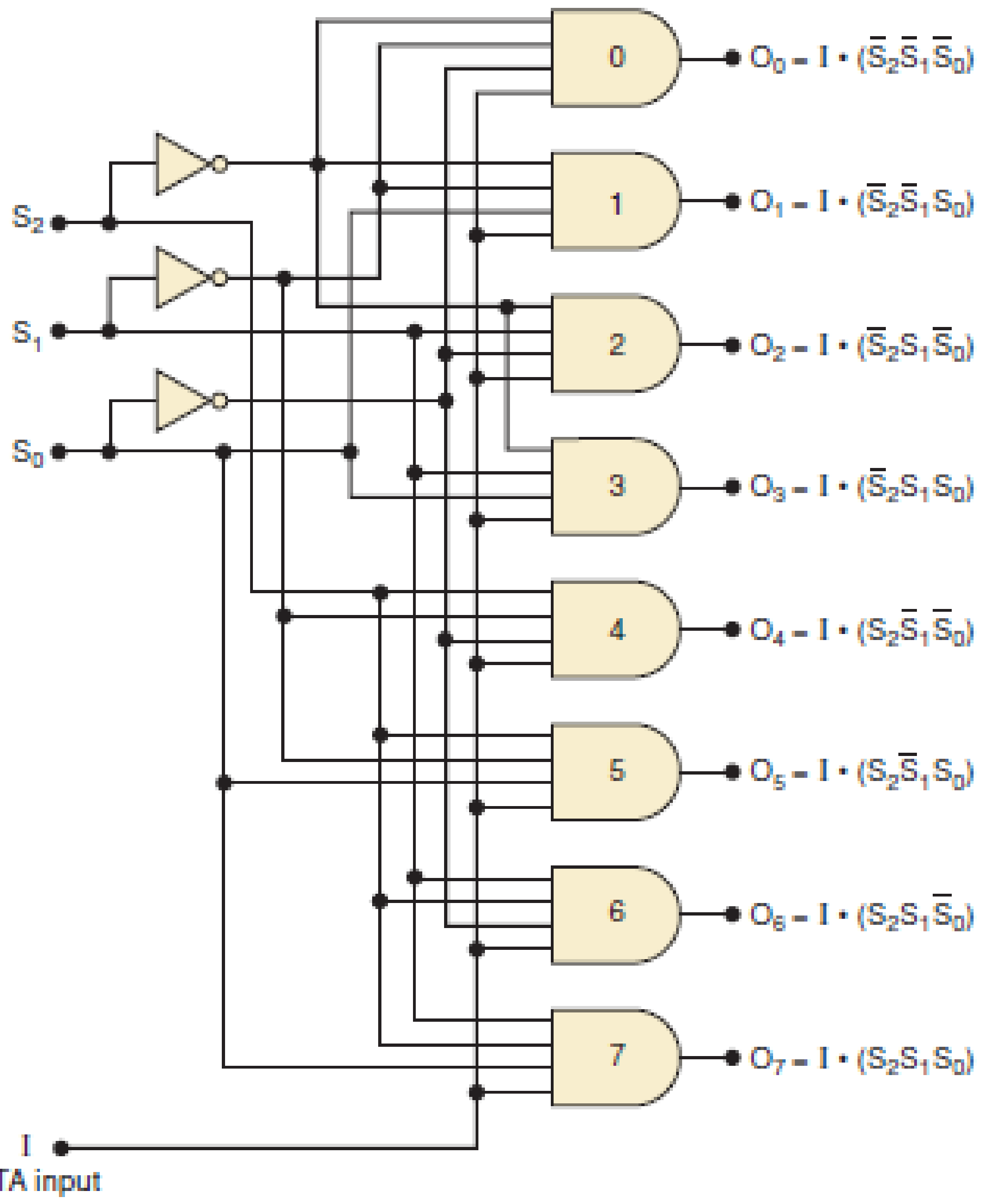

DEMUX
DATA input
O_0
O_1
O_{N-1}
SELECT input
DATA input is transmitted to only one of the outputs as determined by select input code.

1-Line-to-8-Line Demultiplexer

- A demultiplexer that distributes one input line to eight output lines.
- Single data input line *I* is connected to all eight AND gates, but only one of these gates will be enabled by the SELECT input lines.
- EX. With $S_2S_1S_0$ = 000, only AND gate 0 will be enabled and data input *I* will appear at output O_0.
- Other SELECT codes cause input *I* to reach other outputs.

SELECT code			OUTPUTS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

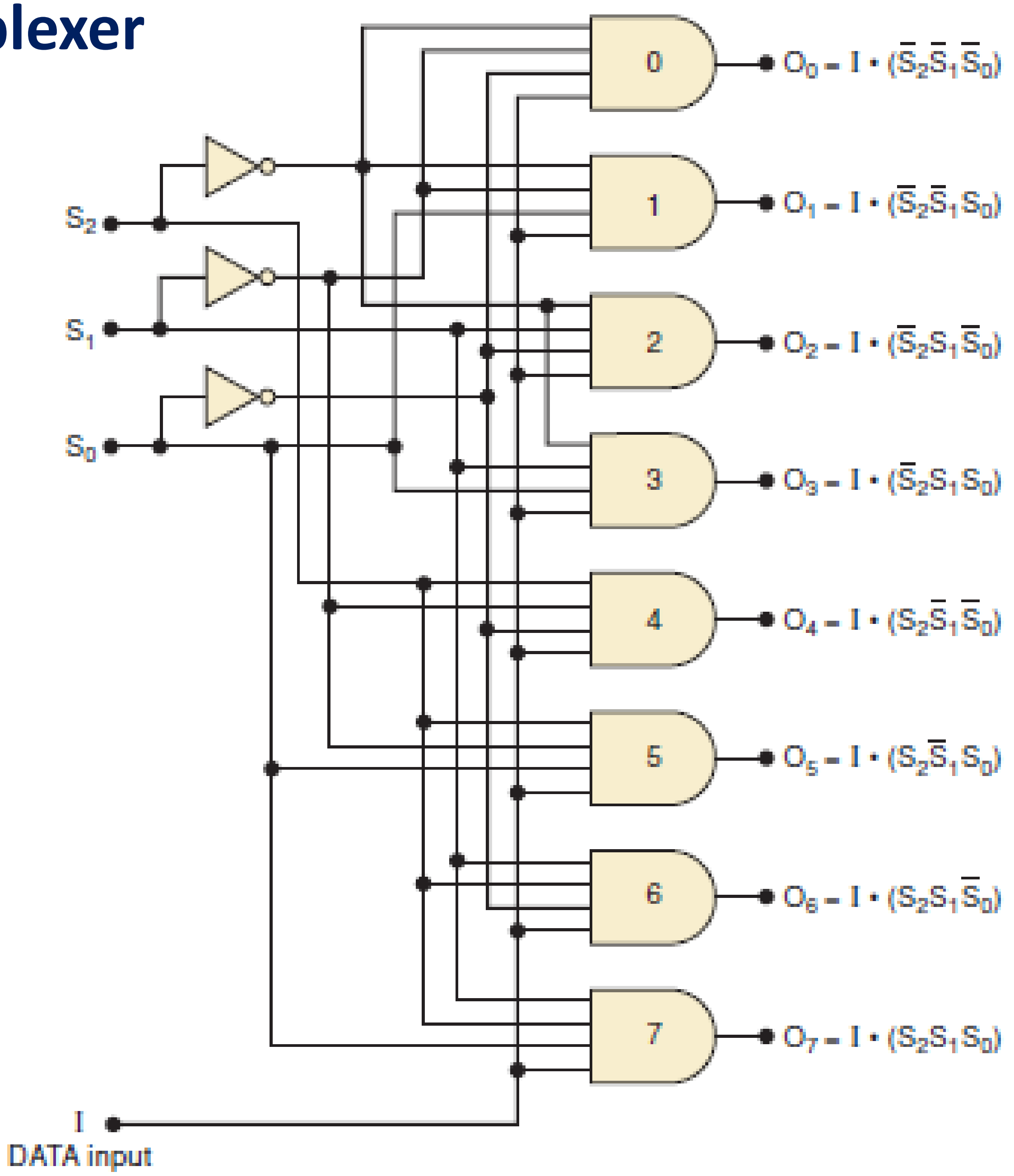
Note: I is the data input



$O_0 = I \cdot (\bar{S}_2\bar{S}_1\bar{S}_0)$
$O_1 = I \cdot (\bar{S}_2\bar{S}_1S_0)$
$O_2 = I \cdot (\bar{S}_2S_1\bar{S}_0)$
$O_3 = I \cdot (\bar{S}_2S_1S_0)$
$O_4 = I \cdot (S_2\bar{S}_1\bar{S}_0)$
$O_5 = I \cdot (S_2\bar{S}_1S_0)$
$O_6 = I \cdot (S_2S_1\bar{S}_0)$
$O_7 = I \cdot (S_2S_1S_0)$
0
1
2
3
4
5
6
7
S_2
S_1
S_0
I
DATA input

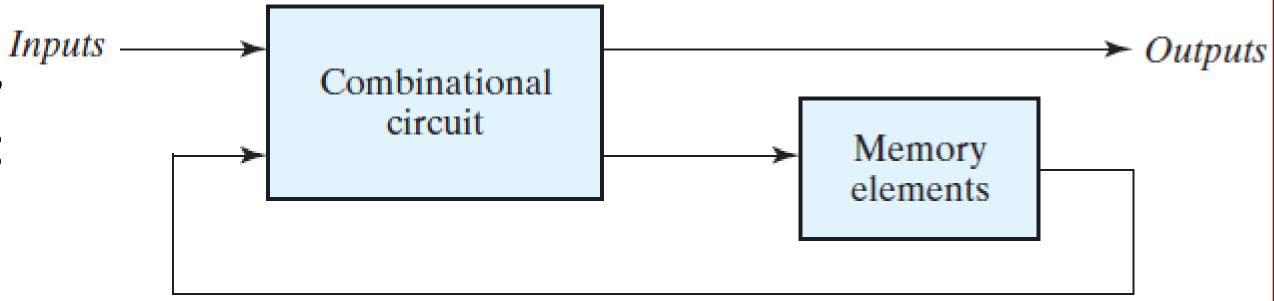
1-Line-to-8-Line Demultiplexer

- Demultiplexer circuit is similar to 3-to-8-line decoder circuit except that a fourth input (*I*) has been added to each gate.
- Many IC decoders have an ENABLE input, which is an extra input added to decoder gates.
- This type of decoder chip can therefore be used as a demultiplexer, with the binary code inputs serving as SELECT inputs and ENABLE input serving as the data input *I.*
- For this reason, IC manufacturers often call this type of device a *decoder/demultiplexer,* and it can be used for either function.


S_2
S_1
S_0
0
1
2
3
4
5
6
7
$O_0 = I \cdot (\bar{S}_2\bar{S}_1\bar{S}_0)$
$O_1 = I \cdot (\bar{S}_2\bar{S}_1 S_0)$
$O_2 = I \cdot (\bar{S}_2 S_1\bar{S}_0)$
$O_3 = I \cdot (\bar{S}_2 S_1 S_0)$
$O_4 = I \cdot (S_2\bar{S}_1\bar{S}_0)$
$O_5 = I \cdot (S_2\bar{S}_1 S_0)$
$O_6 = I \cdot (S_2 S_1\bar{S}_0)$
$O_7 = I \cdot (S_2 S_1 S_0)$
I
DATA input

SEQUENTIAL CIRCUITS

- Sequential circuit receives binary information from external inputs that, together with present state of storage elements, determine binary value of outputs.
- External inputs also determine the condition for changing the state in storage elements.
- Outputs in a sequential circuit are a function not only of the inputs, but also of the present state of the storage elements.
- Next state of the storage elements is also a function of external inputs and present state.
- *A sequential circuit is specified by a time sequence of inputs, outputs, and internal states .*
- In contrast, outputs of combinational logic depend only on the present values of the inputs.
- Two main types of sequential circuits, function of timing of their signals: *synchronous* and *asynchronous*



SEQUENTIAL CIRCUITS

- **A synchronous sequential circuit**: A system whose behaviour can be defined from knowledge of its signals at discrete instants of time.
- **An asynchronous sequential circuit**: A system whose behaviour depends upon input signals at any instant of time and order in which the inputs change.
- Storage elements used in asynchronous sequential circuits are time-delay devices.
- Storage capability of time-delay device varies with time it takes for signal to propagate through device.
- Internal propagation delay of logic gates is sufficient to produce needed delay, so that actual delay units may not be necessary.
- In gate-type asynchronous systems, storage elements consist of logic gates whose propagation delay provides required storage.
- An asynchronous sequential circuit is a combinational circuit with feedback.
- Because of feedback among logic gates, an asynchronous sequential circuit may become unstable at times hence difficult to design.

SEQUENTIAL CIRCUITS

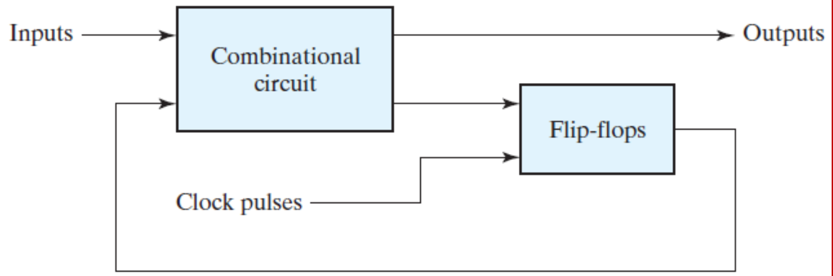
- A synchronous sequential circuit employs signals that affect the storage elements at only discrete instants of time.
- Synchronization is achieved by a timing device called a clock generator, which provides a clock signal having form of a periodic train of clock pulses, denoted by identifiers clock and clk.
- Clock pulses are distributed throughout system in such a way that storage elements are affected only with arrival of each pulse.
- Clock pulses determine when computational activity will occur within the circuit, and other signals (external inputs and otherwise) determine what changes will take place affecting the storage elements and the outputs.
- For example, a circuit that is to add and store two binary numbers would compute their sum from the values of the numbers and store the sum at the occurrence of a clock pulse.

SEQUENTIAL CIRCUITS

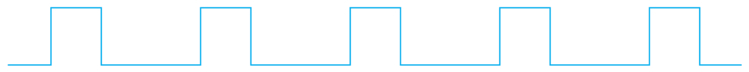
- Synchronous sequential circuits that use clock pulses to control storage elements are called *clocked sequential circuits* and are most frequently used.
- They are called synchronous circuits because the activity within the circuit and the resulting updating of stored values is synchronized to the occurrence of clock pulses.
- Design of synchronous circuits is feasible because they seldom manifest instability problems and their timing is easily broken down into independent discrete steps, each of which can be considered separately.
- Storage elements (memory) used in clocked sequential circuits are called flipflops.
- A flip-flop is a binary storage device capable of storing one bit of information.
- In a stable state, output of a flip-flop is either 0 or 1.
- A sequential circuit may use many flip-flops to store as many bits as necessary.

SYNCHRONOUS CLOCKED SEQUENTIAL CIRCUIT

- Outputs are formed by a combinational logic function of inputs to circuit or values stored in flip-flops (or both).
- Value that is stored in a flip-flop when clock pulse occurs is also determined by inputs to circuit or values presently stored in flip-flop (or both).
- New value is stored (i.e., flip-flop is updated) when a pulse of clock signal occurs
- Prior to the occurrence of clock pulse, combinational logic forming the next value of flip-flop must have reached a stable value.
- Speed at which combinational logic circuits operate is critical.



(a) Block diagram

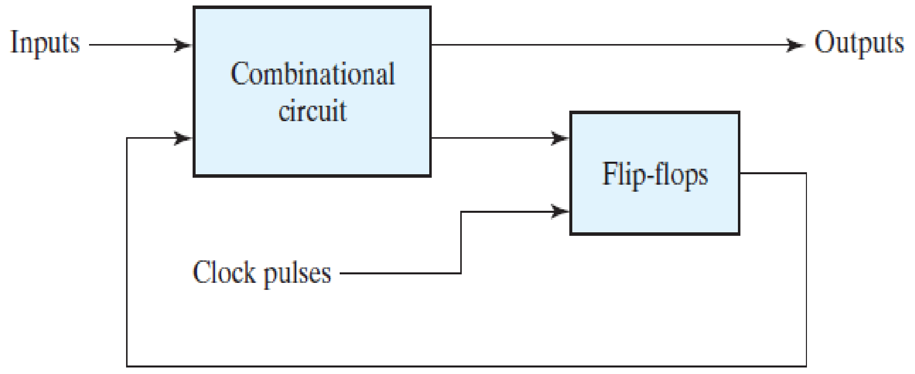


(b) Timing diagram of clock pulses

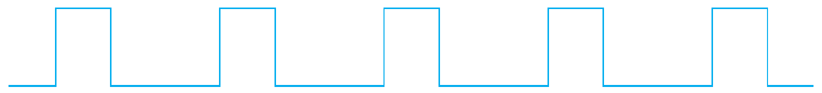
- If clock (synchronizing) pulses arrive at a regular interval, combinational logic must respond to a change in state of flip-flop in time to be updated before the next pulse arrives.
- Propagation delays play an important role in determining minimum interval between clock pulses that will allow the circuit to operate correctly.

SYNCHRONOUS CLOCKED SEQUENTIAL CIRCUIT

- A change in state of flip-flops is initiated only by a clock pulse transition—for example, when value of clock signals changes from 0 to 1.
- When a clock pulse is not active, feedback loop between the value stored in the flip-flop and value formed at input to the flip-flop is effectively broken.
- Flipflop outputs cannot change even if the outputs of the combinational circuit driving their inputs change in value.
- Thus, transition from one state to the next occurs only at predetermined intervals dictated by the clock pulses.



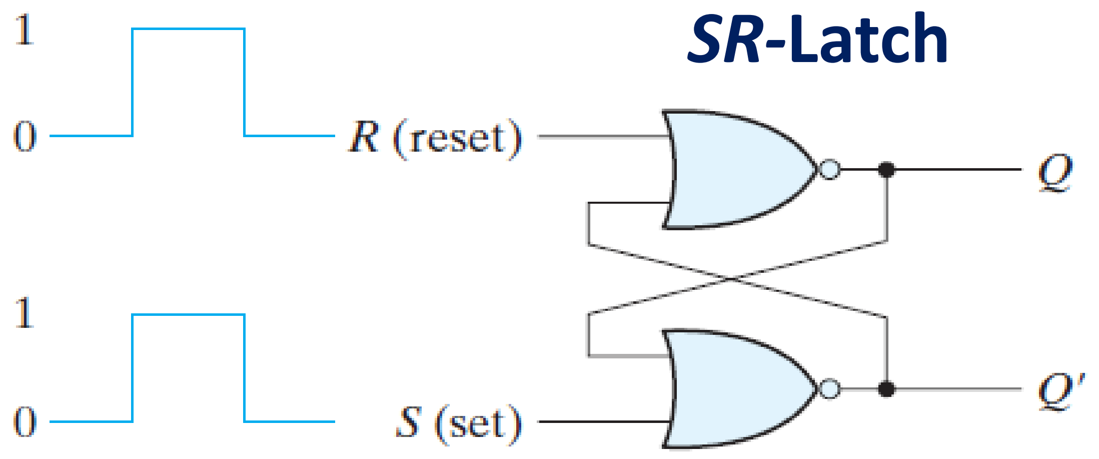
(a) Block diagram


(b) Timing diagram of clock pulses

STORAGE ELEMENTS: LATCHES

- A storage element in a digital circuit can maintain a binary state indefinitely (as long as power is delivered to the circuit), until directed by an input signal to switch states.
- Major differences among various types of storage elements are in the number of inputs they possess and in the manner in which the inputs affect the binary state.
- *Storage elements that operate with signal levels (rather than signal transitions) are referred to as latches* ; *those controlled by a clock transition are flip-flops*.
- Latches are level sensitive devices; flip-flops are edge-sensitive devices.
- Latches are basic circuits from which all flip-flops are constructed.
- Latches are useful for storing binary information and for the design of asynchronous sequential circuits but not practical for use as storage elements in synchronous sequential circuits.

SR-Latch



(a) Logic diagram

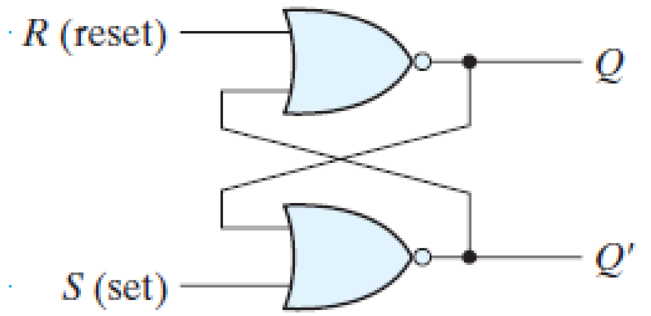
S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

- *SR* latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates.
- When output *Q* = 1 and *Q* = 0, the latch is said to be in *set state* .
- When *Q* = 0 and *Q* = 1, it is in *reset state* .
- Outputs *Q* and *Q* are normally complement of each other.
- When both inputs are equal to 1 at the same time, a condition in which both outputs are equal to 0 (rather than be mutually complementary) occurs.
- If both inputs are then switched to 0 simultaneously, device will enter an unpredictable or undefined state or a metastable state.

SR-Latch

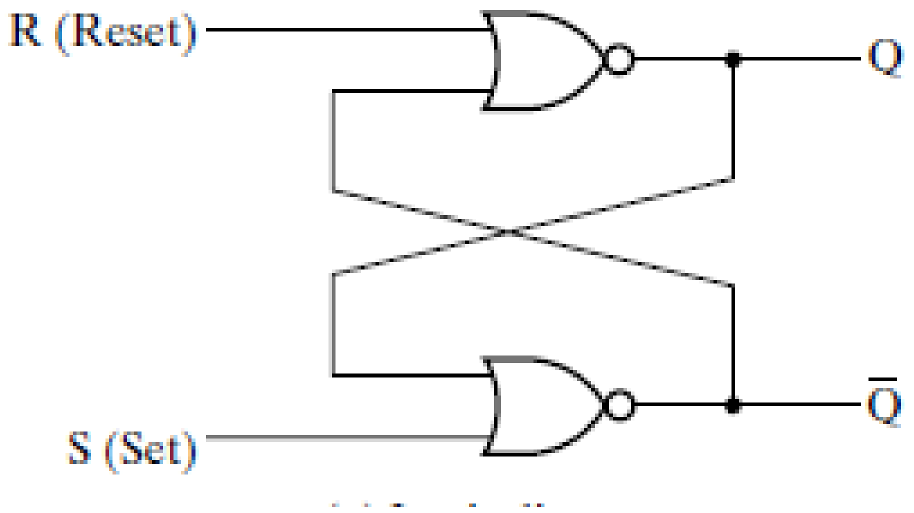
- Under normal conditions, both inputs of latch remain at 0 unless state has to be changed.
- Application of a momentary 1 to *S* input causes latch to go to *set* state. (S = 1, R = 0)
- *S* input must go back to 0 before any other changes take place, in order to avoid occurrence of an undefined next state that results from forbidden input condition.
- Two input conditions cause circuit to be in the *set* state.
- Removing active input from S leaves the circuit in the same state.
- (S = 1, R = 0, Q =1, Q' =0)
- It is then possible to shift to the *reset* state by momentary applying a 1 to R input.
- Removing active input from R leaves the circuit in same state.
- *When both inputs S and R are equal to 0, latch can be in either set or reset state, depending on which input was most recently a 1.*



S	R	Q	Q'	
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)

SR-Latch

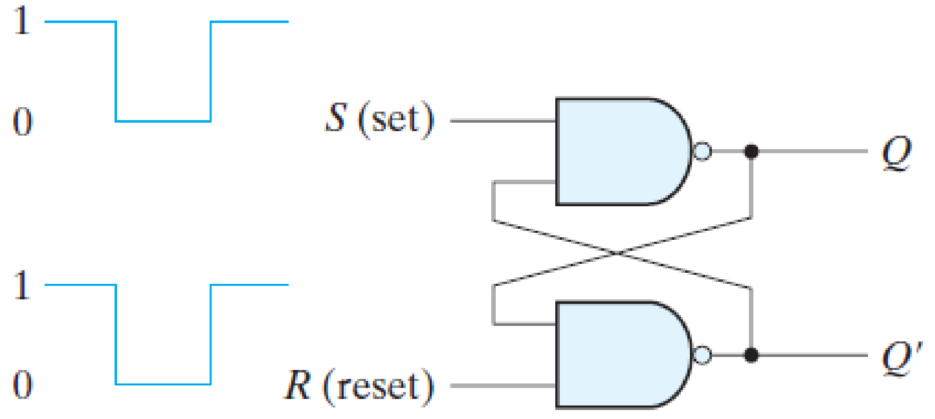
- If a 1 is applied to both *S* and *R* inputs of the latch, both outputs go to 0.
- This action produces an undefined next state, because the state that results from the input transitions depends on the order in which they return to 0.
- It also violates the requirement that outputs be the complement of each other.
- In normal operation, this condition is avoided by making sure that 1's are not applied to both inputs simultaneously.

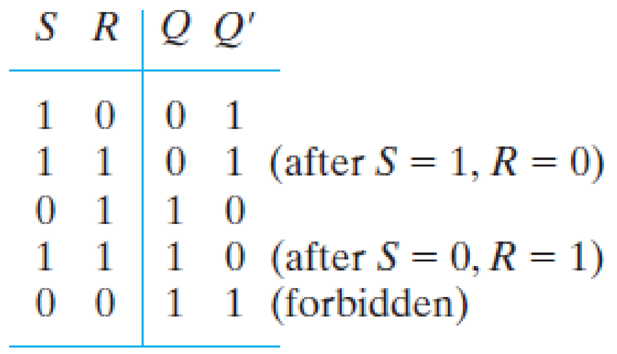


S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

SR-Latch

- *SR* latch can also be constructed with two cross-coupled NAND gates.
- It operates with both inputs normally at 1, unless the state of the latch has to be changed.
- Application of 0 to *S* input causes output *Q* to go to 1, latch is in *set* state.
- When *S* input goes back to 1, circuit remains in the *set* state.
- After both inputs go back to 1, state of the latch can be changed by R= 0.
- This causes circuit to go to *reset* state and stay there even after both inputs return to 1.
- When both inputs equal to 0 at the same time, condition is forbidden for NAND latch.
- This input combination that should be avoided.



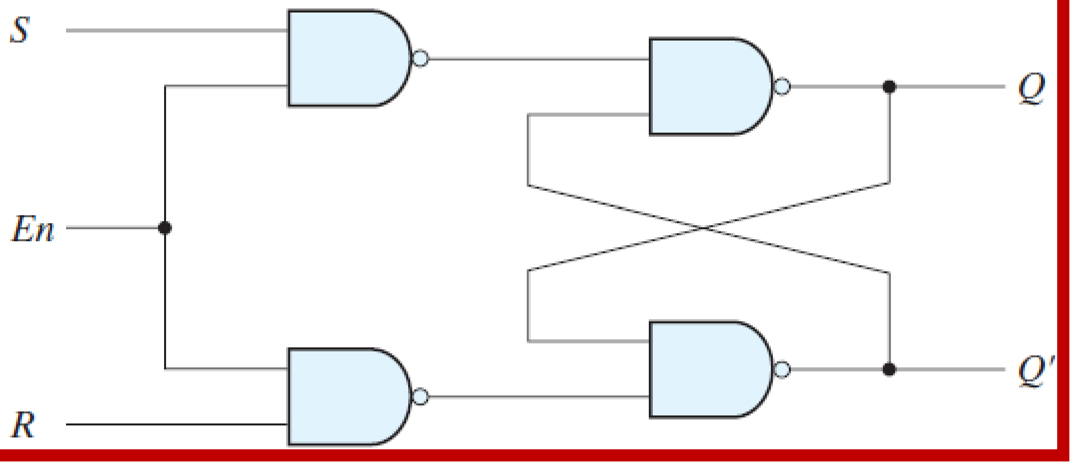


S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

SR-LATCH WITH A CONTROL INPUT

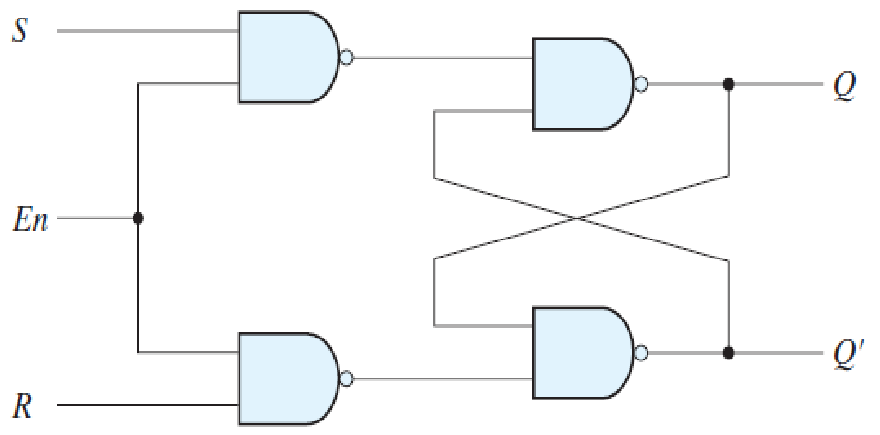
- Input signals for NAND require complement of those values used for NOR latch.
- Since NAND latch requires a 0 signal to change its state, it is referred as an *S'R'* latch.
- Operation of basic *SR* latch can be modified by providing an additional input.
- Signal that determines (controls) *when* the state of latch can be changed by determining whether *S* and *R* (or *S'* and *R'*) can affect circuit.
- It consists of basic *SR* latch and two additional NAND gates.
- Control input E_n acts as an *enable* signal for other two inputs.
- **Outputs of NAND gates stay at logic-1 level as long as enable signal remains at 0.**

En	*S*	*R*	Next state of *Q*
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate



SR-LATCH WITH A CONTROL INPUT

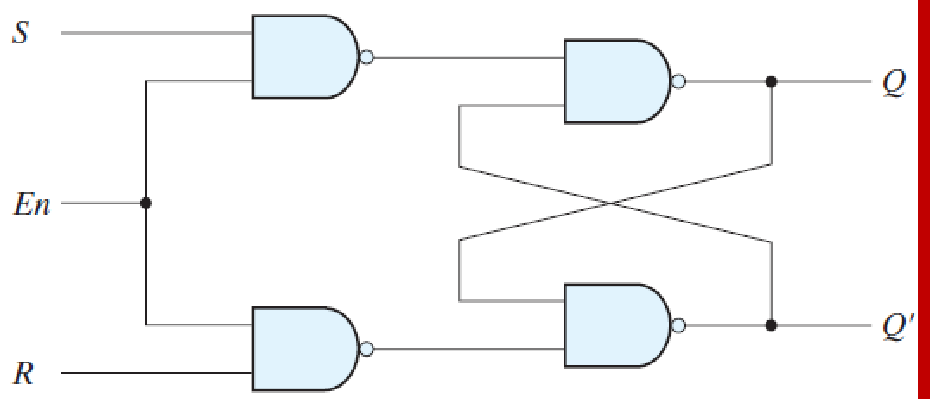
- When E_n goes to 1, information from *S* or *R* input is allowed to affect latch.
- The *set* state is reached with *S* = 1, *R* = 0, and E_n = 1 (active-high enabled).
- To change to *reset* state, inputs must be *S* = 0, *R* = 1, and E_n = 1.
- In either case, when E_n returns to 0, circuit remains in its current state.
- Control input disables circuit by applying 0 to E_n, so that the state of the output does not change regardless of the values of *S* and *R* .
- Moreover, when E_n = 1 and both *S* and *R* inputs are equal to 0, the state of the circuit does not change.



En	*S*	*R*	Next state of *Q*
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

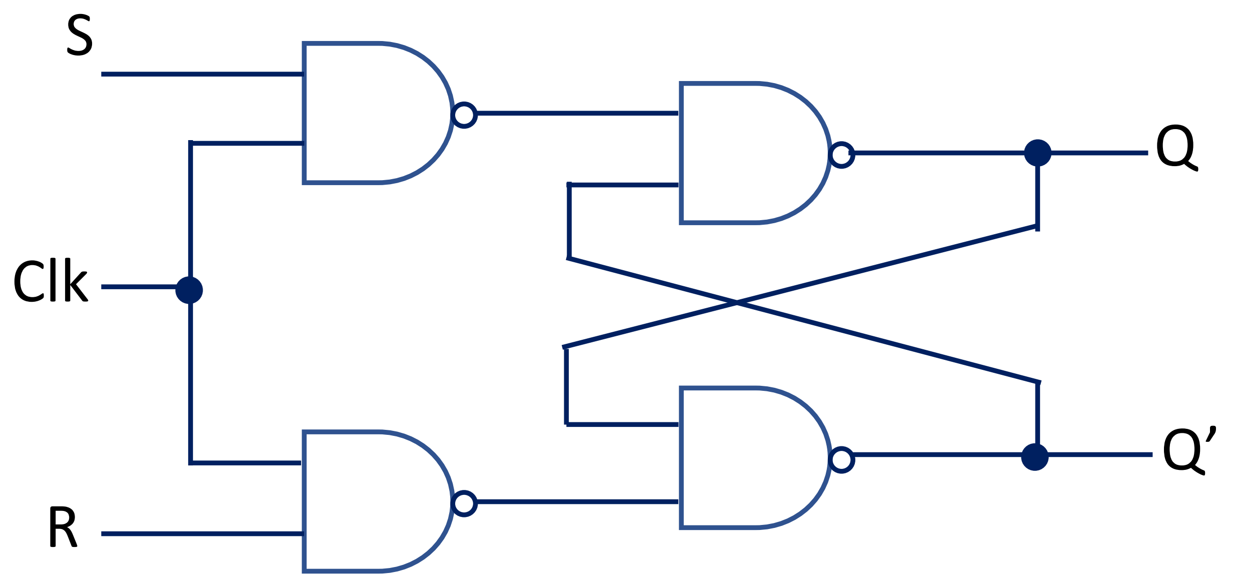
SR-LATCH WITH A CONTROL INPUT

- An indeterminate condition occurs when all three inputs are equal to 1.
- This condition places 0's on both inputs of basic *SR* latch, which puts it in *undefined state*.
- When E_n goes back to 0, one cannot conclusively determine next state, because it depends on whether *S* or *R* input goes to 0 first.
- This indeterminate condition makes circuit difficult to manage, and it is seldom used in practice.
- Nevertheless, *SR* latch is an important circuit because other useful latches and flip-flops are constructed from it.

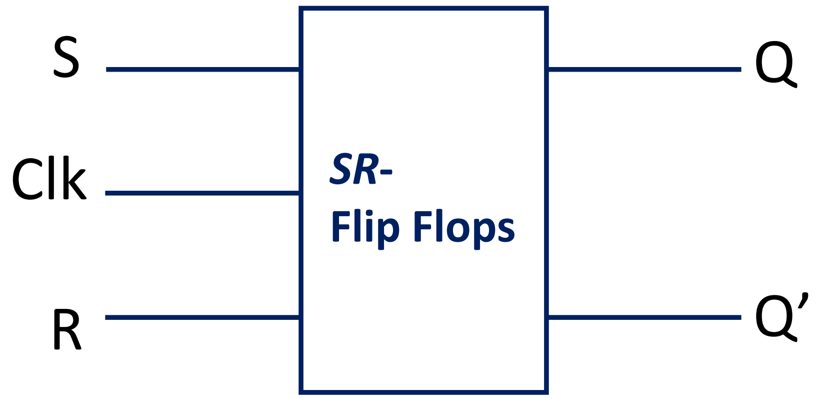


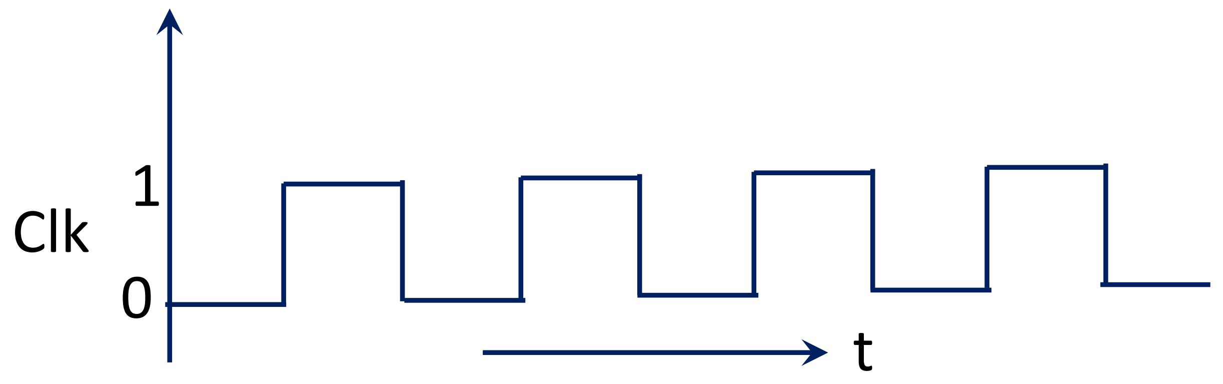
En	*S*	*R*	Next state of *Q*
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

SR-Flip Flops



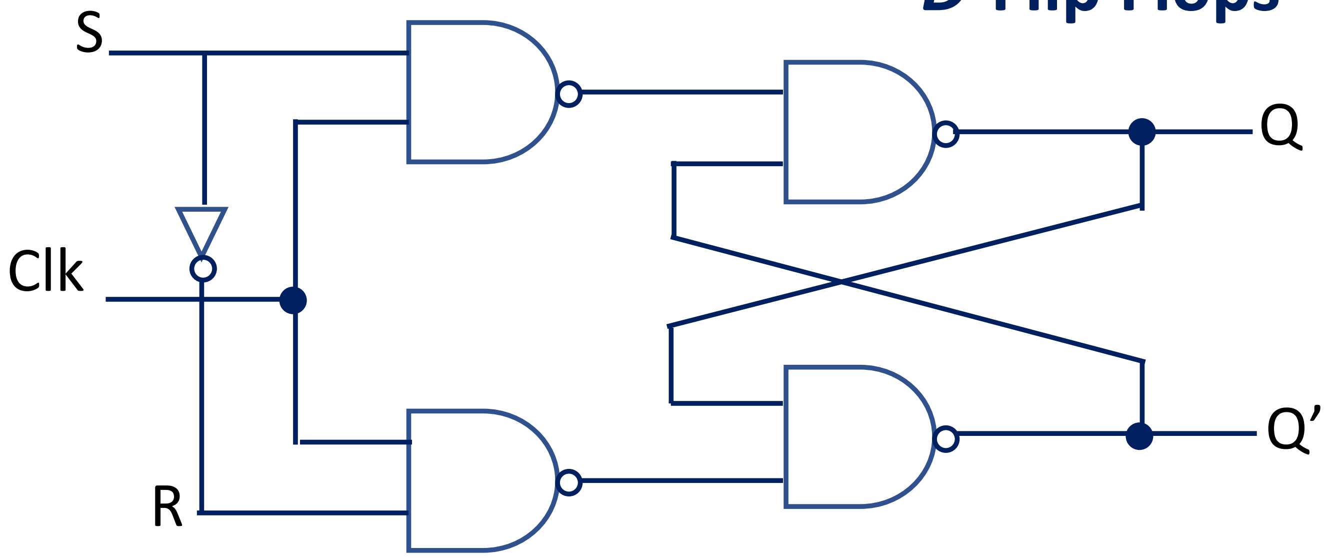
Clk	S	R	Q	Q’
0	X	X	Memory	
1	0	0	Memory	
1	1	0	1	0
1	0	1	0	1
1	1	1	Not used	



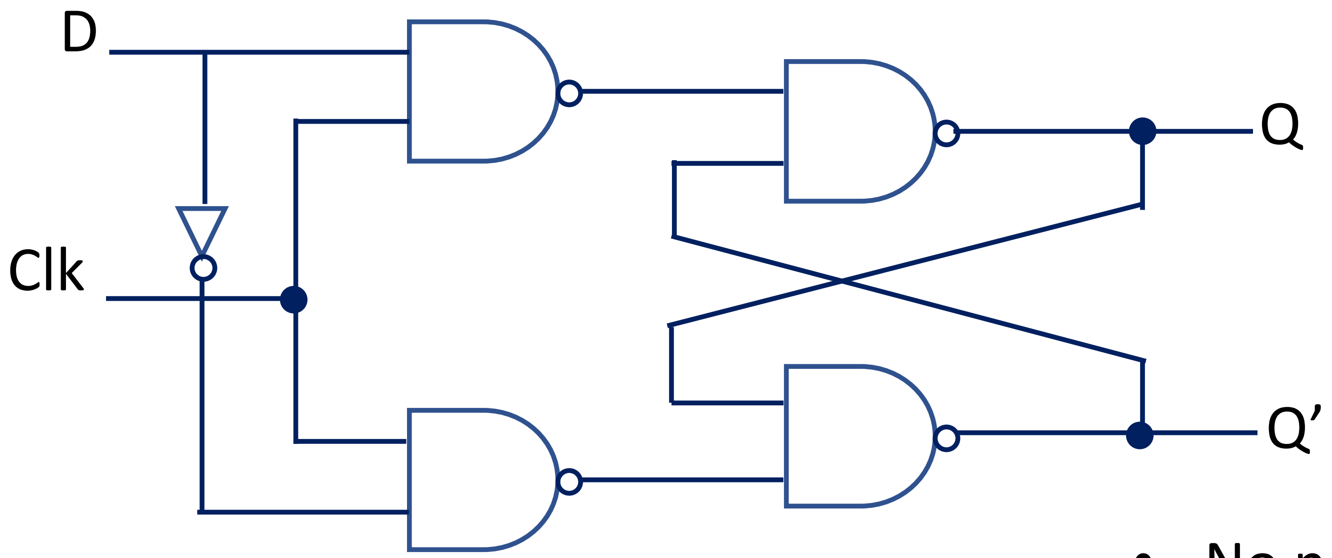


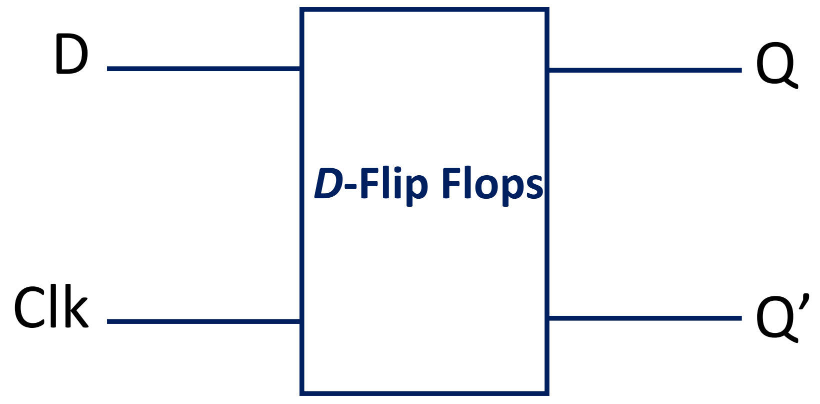
- Inputs are applied when clock is 0 and then enable the clock to transfer the inputs to output.

D-Flip Flops



Clk	D	Q	Q’
0	X	Memory	
1	1	1	0
1	0	0	1
1	1	Not used	





- No need to put S = 0 and R = 0 for memory.