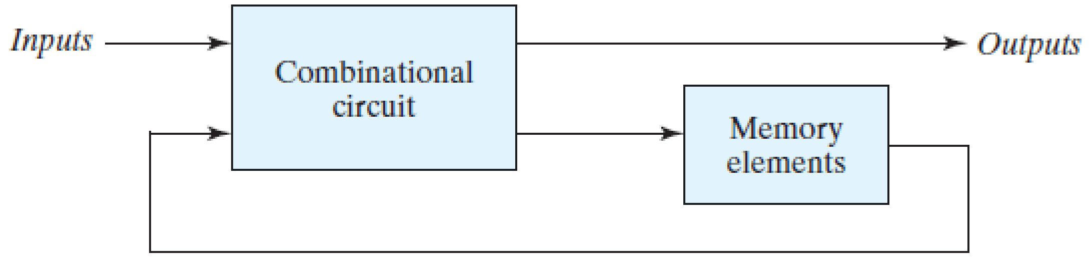
LOGIC DESIGN EET-1021

CHAPTER 05

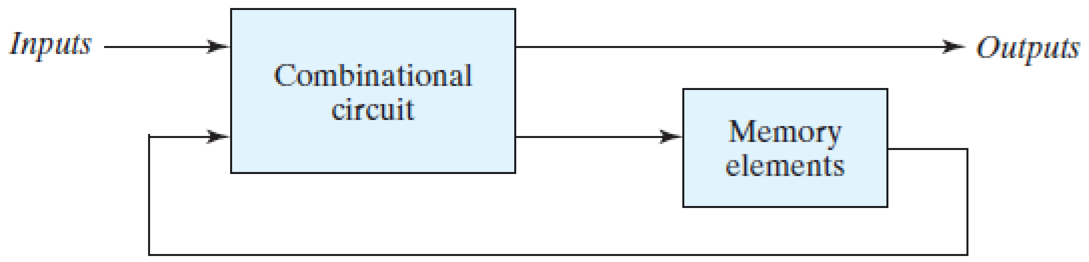
Lecture 25

SEQUENTIAL LOGIC CIRCUITS

Block diagram of sequential circuit



- It consists of a combinational circuit to which storage elements are connected to form a feedback path.
- The storage elements are devices capable of storing binary information.
- The binary information stored in these elements at any given time defines the *state of the sequential circuit at* that time.
- The sequential circuit receives binary information from external inputs that, together with the present state of the storage elements, determine the binary value of the outputs.



- These external inputs also determine the condition for changing the state in the storage elements.
- The block diagram demonstrates that the outputs in a sequential circuit are a function not only of the inputs, but also of the present state of the storage elements.
- The next state of the storage elements is also a function of external inputs and the present state.
- Thus, **a sequential circuit is specified by a time sequence of inputs, outputs, and internal states** .
- In contrast, **the outputs of combinational logic depend only** on the present values of the inputs.

Classification of sequential circuit

- There are two main types of sequential circuits, and their classification is a function of the timing of their signals.

- A *synchronous sequential circuit is a system whose behavior* can be defined from the knowledge of its signals at discrete instants of time.

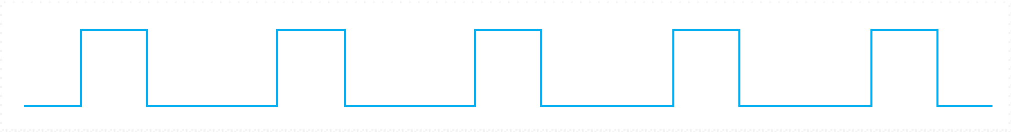
- The behavior of an *asynchronous sequential circuit depends upon the input signals at any instant of time and the order in which the inputs change.*

Asynchronous sequential circuit

- The storage elements commonly used in asynchronous sequential circuits are time-delay devices.
- The storage capability of a time-delay device varies with the time it takes for the signal to propagate through the device. In practice, the internal propagation delay of logic gates is of sufficient duration to produce the needed delay, so that actual delay units may not be necessary.
- In gate-type asynchronous systems, the storage elements consist of logic gates whose propagation delay provides the required storage.
- Thus, an asynchronous sequential circuit may be regarded as a combinational circuit with feedback. Because of the feedback among logic gates, an asynchronous sequential circuit may become unstable at times.

Synchronous sequential circuit

- 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 the form of a periodic train of clock pulses* .
- *The clock signal is commonly denoted by the identifiers **clock** and **clk** .*
- *The clock* pulses are distributed throughout the system in such a way that storage elements are affected only with the arrival of each pulse.
- In practice, the 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.



Timing diagram of clock pulses

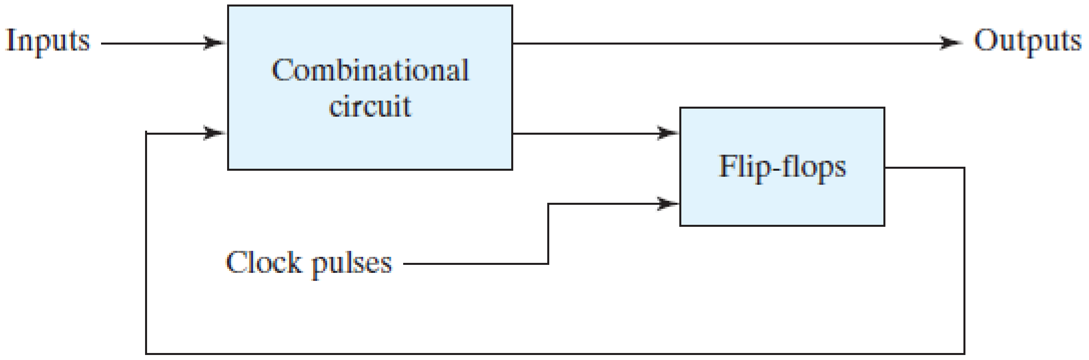
- For example, a circuit that is to add and store two binary numbers would compute their sum from the values of the numbers and store the sum at the **occurrence of a clock pulse**.

- Synchronous sequential circuits that use clock pulses to control storage elements are called *clocked sequential circuits and are the type* ***most frequently*** *encountered* in practice.

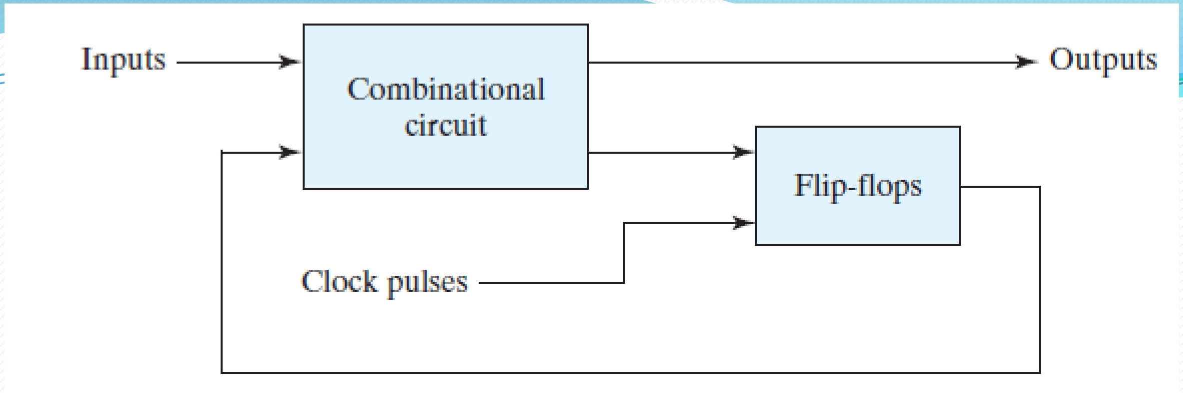
- They are called *synchronous circuits because the activity within the* circuit and the resulting updating of stored values is synchronized to the occurrence of clock pulses.

- The 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.

Block diagram of Synchronous clocked sequential circuit



- The 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, the 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.
- The *outputs are formed by a combinational logic* function of the inputs to the circuit or the values stored in the flip-flops (or both).



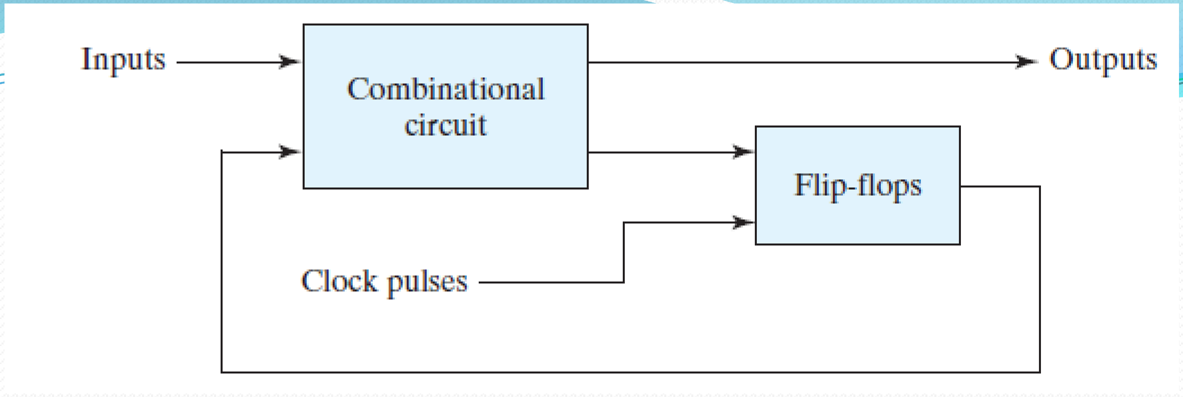
- The value that is stored in a flip-flop when the clock pulse occurs is also determined by the inputs to the circuit or the values presently stored in the flip-flop (or both).

- The new value is stored (i.e., the flip-flop is updated) when a pulse of the clock signal occurs.

- Prior to the occurrence of the clock pulse, the combinational logic forming the next value of the flip-flop must have reached a stable value. Consequently, the speed at which the combinational logic circuits operate is critical.

- If the clock (synchronizing) pulses arrive at a regular interval, the combinational logic must respond to a change in the state of the flip-flop in time to be updated before the next pulse arrives.

- **Propagation delays** play an important role in determining the minimum interval between clock pulses that will allow the circuit to operate correctly.



- A change in state of the flip-flops is initiated only by a clock pulse transition—for example, when the value of the clock signals changes from 0 to 1.

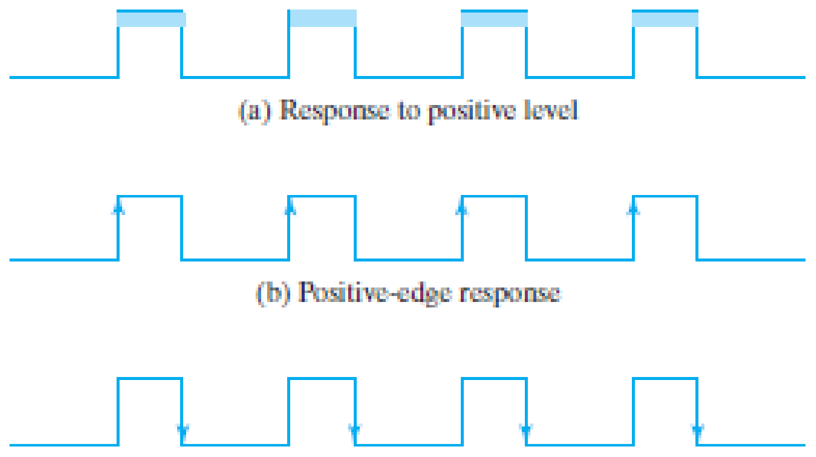
- When a clock pulse is not active, the feedback loop between the value stored in the flip-flop and the value formed at the input to the flip-flop is effectively broken because the flipflop outputs cannot change even if the outputs of the combinational circuit driving their inputs change in value.

- Thus, the transition from one state to the next occurs only at predetermined intervals dictated by the **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.
- The 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 said to be level* sensitive devices; flip-flops are edge-sensitive devices.**

flop

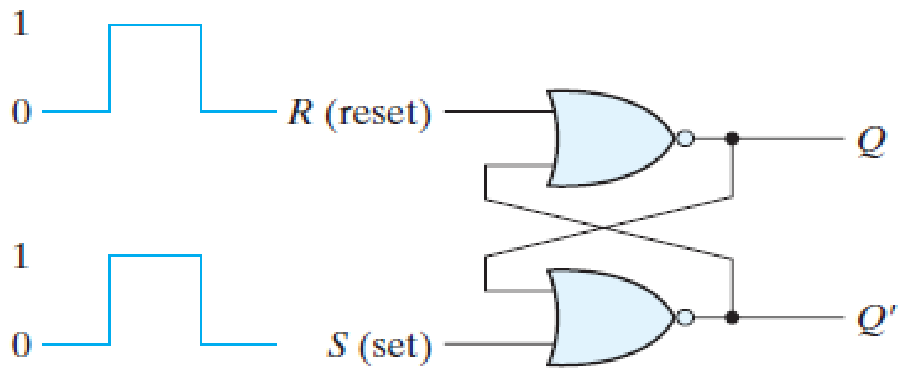

(a) Response to positive level

(b) Positive-edge response

(c) Negative-edge response

- The latch responds to a change in the ***level*** *of a clock pulse.*
- The key to the proper operation of a flip-flop is to trigger it only during a signal *transition .*
- *This can be accomplished* by eliminating the feedback path that is inherent in the operation of the sequential circuit using latches.
- A clock pulse goes through two transitions: from 0 to 1 and the return from 1 to 0.
- As shown in Figure , the positive transition is defined as the positive edge and the negative transition as the negative edge.

SR Latch with NOR gates



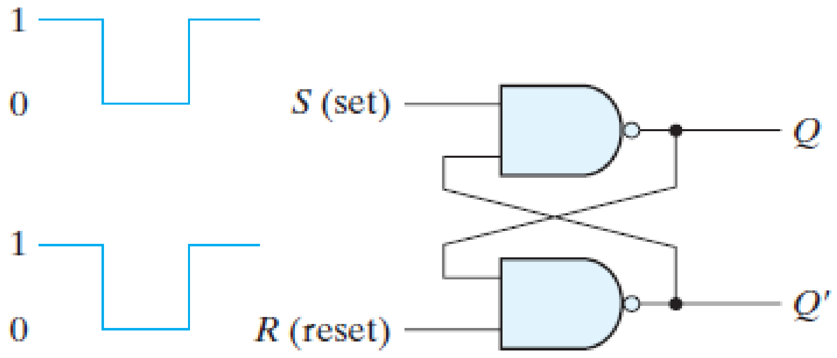
(a) Logic diagram

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

The *SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND*
gates, and two inputs labeled *S for set and R for reset.*
The latch has **two usef**Q' **states**. Q'
When output Q = 1 *and* Q = 0*, the latch is said to be in the* ***set state*** . *When* Q = 0
and Q = 1*, it is* in the ***reset state*** .

SR latch with NAND gates



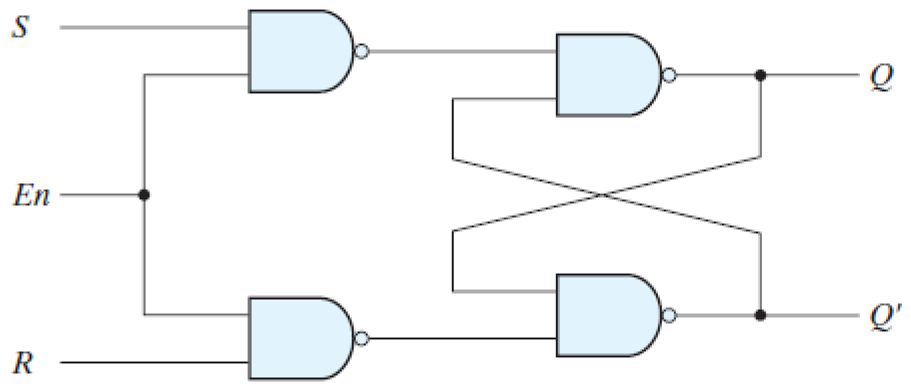
(a) Logic diagram

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)

(b) Function table

•It operates with both inputs normally at 1, unless the state of the latch has to be changed.
•The application of 0 to the *S input causes output Q to go to 1, putting the latch in the set state. When the S* input goes back to 1, the circuit remains in the set state.
•After both inputs go back to 1, we are allowed to change the state of the latch by placing a 0 in the *R input. This action causes* the circuit to go to the reset state and stay there even after both inputs return to 1.
• The condition that is forbidden for the NAND latch is **both inputs being equal to 0** at the same time, an input combination that should be avoided.

SR latch with control input



(a) Logic diagram

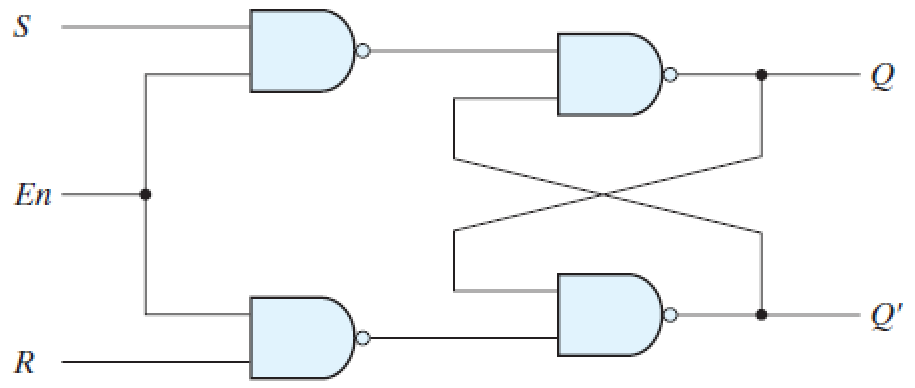
En	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(b) Function table

- It consists of the basic SR latch and two additional NAND gates. **The control input En acts as an enable signal for the other two inputs.**

- **The outputs of the NAND gates stay at the logic-1 level as long as the enable signal remains at 0. This is the quiescent** condition for the SR latch.

- When the enable input goes to 1, information from the S or R input is allowed to affect the latch. **The set state is reached with S = 1, R = 0, and En = 1(active-high enabled).**



(a) Logic diagram

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

(b) Function table

- To change to the **reset state, the inputs must be *S* = 0, *R* = 1, *and En* = 1.** *In either case, when En returns to 0, the circuit remains in its current state.*

- *The* control input disables the circuit by applying 0 to *En, so that the state of the output does* not change regardless of the values of *S and R* .

- *Moreover, when En = 1 and both the S* and *R inputs are equal to 0, the state of the circuit does not change.*

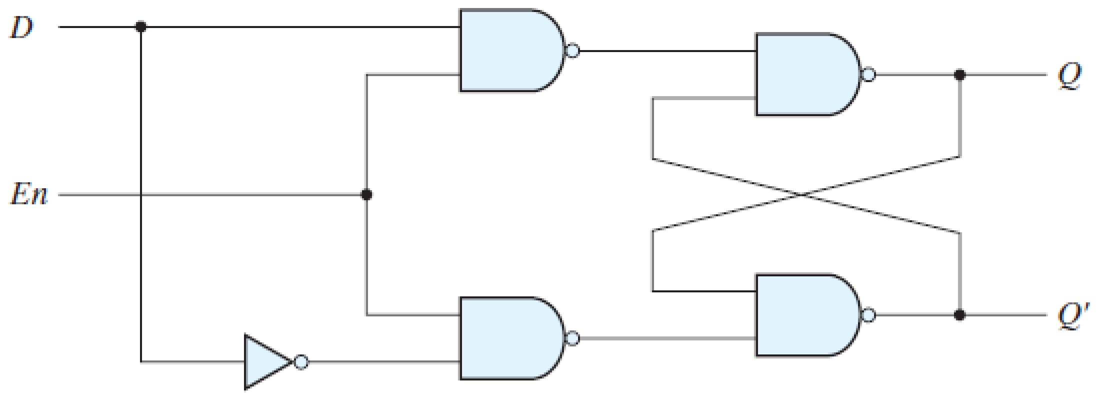
- **An indeterminate condition occurs when all three inputs are equal to 1.**

```
module set-reset (Q, Q', S, R, EN);
input S, R, EN;
output Q, Q';
wire a, b;
nand n1(a, S, EN),
        n2(b, R, EN),
        n3(Q, a, Q'),
        n4(Q', b, Q);



endmodule
```

D Latch (Transparent Latch)



(a) Logic diagram

En	D	Next state of Q
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

(b) Function table

➢One way to eliminate the undesirable condition of the indeterminate state in the SR latch is to ensure that inputs **S and R are never equal to 1 at the same time**. This is done in the D latch, shown in Figure .

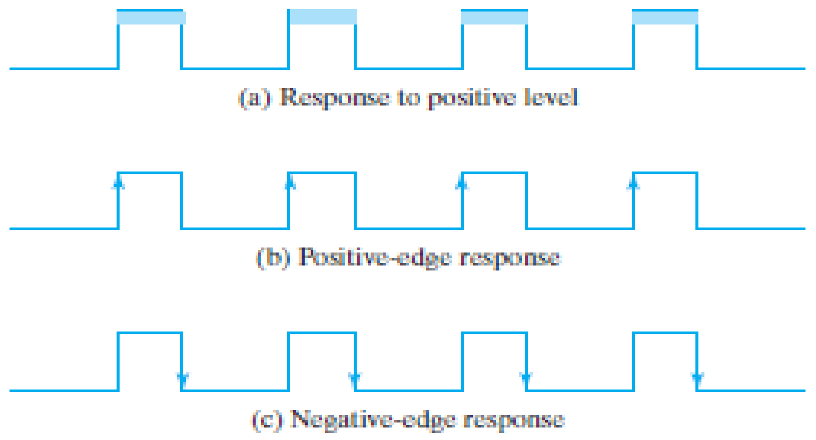
➢This latch has only two inputs: D (data) and En (enable). **The D input goes directly to the S input, and its complement is applied to the R input.**

➢As long as the enable input is at 0, the cross-coupled SR latch has both inputs at the 1 level and the circuit cannot change state regardless of the value of D .

➢The D input is sampled when En = 1. If D = 1, the Q output goes to 1, placing the circuit in **the set state**.

➢ If D = 0, output Q goes to 0, placing the circuit in the **reset state**.

Clock response in latch and flip-flop


(a) Response to positive level
(b) Positive-edge response
(c) Negative-edge response

•The latch responds to a change in the ***level*** *of a clock pulse.*

•The key to the proper operation of a flip-flop is to trigger it only during a signal *transition* .

•*This can be accomplished* by eliminating the feedback path that is inherent in the operation of the sequential circuit using latches.

• A clock pulse goes through two transitions: from 0 to 1 and the return from 1 to 0.

•As shown in Figure , the positive transition is defined as the positive edge and the negative transition as the negative edge.

```
// Description of D latch
 module  D_latch (Q, D, enable);
output  Q;
input  D, enable;
 reg  Q;
 always  @ (enable or D)
  if (enable) Q <= D; // Same as: if (enable == 1)
 endmodule
// Alternative syntax
 module  D_latch (output reg Q,  input  enable, D);
 always  @ (enable, D)
  if  (enable) Q <= D; // No action if enable not asserted
endmodule
```

THANK YOU