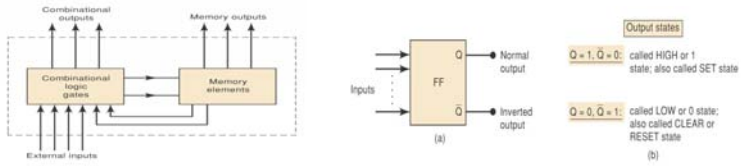
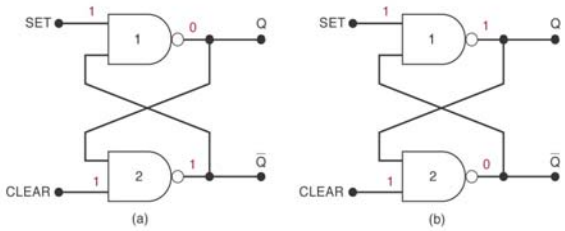
Chapter 5 – Flip Flops Sequential circuits - Introduction



- Logic circuits studied so far have outputs that respond immediately to inputs at some instant in time.
- We now introduce the concept of memory. The flip-flop, abbreviated FF, is a key memory element.
- The outputs of a flip flop are Q and $\overline{Q}$
- Q is understood to be the normal output, $\overline{Q}$ is always the opposite.
- When the normal output (Q) is placed in the high or 1 state we say the FF has been set.
- When the normal output (Q) is placed in the low or 0 state we say the FF has been cleared or reset.

5-1 NAND Gate Latch

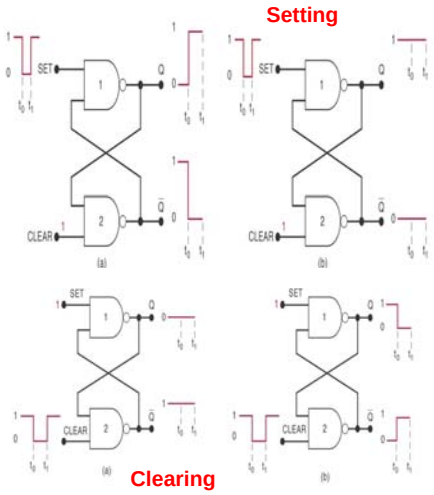


- The NAND gate latch or simply latch is a basic FF.
- The two NAND gates are cross-coupled
- The inputs are set and clear (reset)
- The inputs are active low, that is, the output will change when the input is pulsed low.

5-1 NAND Gate Latch

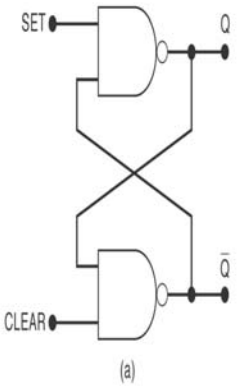
- Setting NAND latch:
  - A low pulse on the SET will always cause the latch to end up in the Q = 1 state. $Q = 1 \text{ and } \overline{Q} = 0$
- Clearing NAND latch:
  - A low pulse on the Clear will always cause the latch to end up in the Q = 0 state.

$$Q = 0 \text{ and } \overline{Q} = 1$$



NAND LATCH

- Summary of the NAND latch:
  - Set = clear = 1. Normal resting state, outputs remain in state prior to input.
  - Set = 0, clear = 1. Q will go high and remain high even if the set input goes high.
  - Set = 1, clear = 0. Q will go low and remain low even if the clear input goes high.
  - Set = clear = 0. Output is unpredictable because the latch is being set and cleared at the same time.

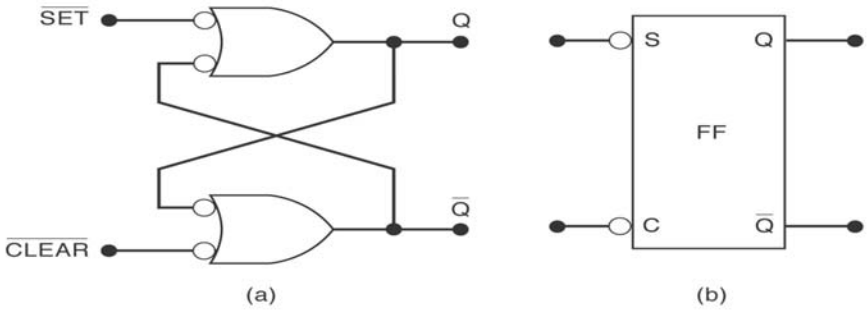


Set	Clear	Output
1	1	No change
0	1	Q = 1
1	0	Q = 0
0	0	Invalid*

*Produces $Q = \overline{Q} = 1$.

(b)

NAND latch equivalent representation; (b) simplified block symbol



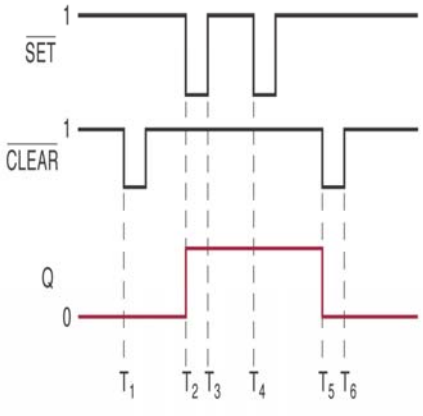
Alternate symbol of NAND latch, The S and C labels represent the SET and CLEAR inputs.

The bubbles indicate the active-low nature of these inputs

Clear and Reset are interchangeable terms.

Example 5-1

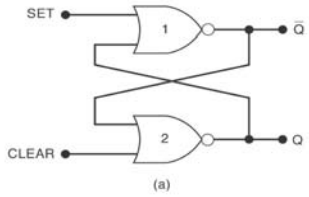
- Waveforms are applied at the NAND latch:
  - Assume that initially Q=0, determine the Q waveform.
    - SET=CLEAR=1, no change
    - At T1, low pulse on CLEAR has no effect since Q=0.
    - At T2, low pulse on SET will cause Q=1 and remains high when SET goes high at T3.
    - At T4, low pulse on SET will have no effect since Q=1.
    - At T5, low pulse on CLEAR will cause Q=0 and remains low when CLEAR goes high at T6



5-2 NOR Gate Latch

- The NOR latch is similar to the NAND latch except that the Q and $\overline{Q}$ outputs are reversed.
- The set and clear inputs are active high, that is, the output will change when the input is pulsed high.
- In order to ensure that a FF begins operation at a known level, a pulse may be applied to the set or clear inputs when a device is powered up.

NOR gate latch: truth table and simplified block symbol

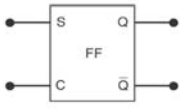


(a)

Set	Clear	Output
0	0	No change
1	0	Q = 1
0	1	Q = 0
1	1	Invalid*

*Produces Q = Q̄ = 0.

(b)

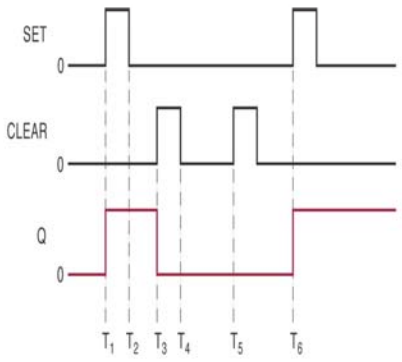


(c)

- Summary of the NAND latch:
  - Set = clear = 0. Normal resting state, outputs remain in state prior to input.
  - Set = 1, clear = 0. Q will go high and remain high even if the set input goes low.
  - Set = 0, clear = 1. Q will go low and remain low even if the clear input goes low.
  - Set = clear = 1. Output is unpredictable because the latch is being set and cleared at the same time.

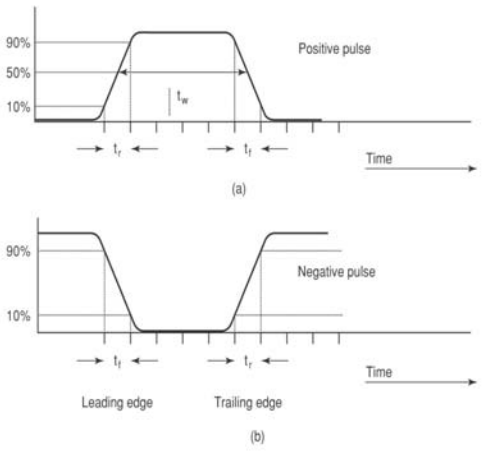
Example 5-3

- Waveforms are applied at the NOR latch:
  - Assume that initially Q=0, determine the Q waveform.
    - SET=CLEAR=0, no change
    - At T1, high pulse on SET causes Q to go high and remain high
    - At T2, low pulse on SET will cause no effect on Q.
    - At T3, high pulse on CLEAR will clear Q, Q=0 and remains low even after CLEAR return low at T4.
    - At T5, high pulse on CLEAR will have no effect on Q
    - St T6, a high pulse on SET causes Q to go back High and stays high



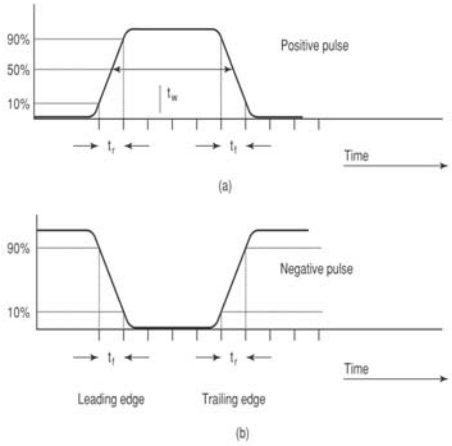
5-4 Digital Pulses

- Signals that switch between active and inactive states are called pulse waveforms.
  - A positive pulse has an active high level.
  - A negative pulse has an active low level.



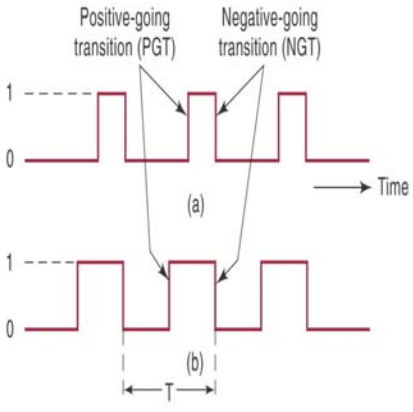
5-4 Digital Pulses

- The transition from low to high on a positive pulse is called rise time (t_r).
  - Rise time is measured between the 10% and 90% points on the leading edge of the voltage waveform.
- The transition from high to low on a positive pulse is called fall time (t_f).
  - Fall time is measured between the 90% and 10% points on the trailing edge of the voltage waveform.
- The pulse width (t_w) is defined as the time between the points when the leading and trailing edges are at 50% of the high level.



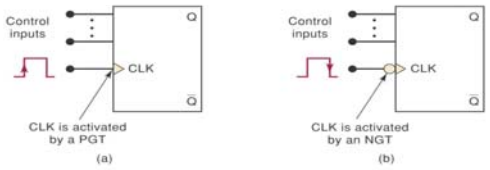
5-5 Clock Signals and Clocked Flip-Flops

- Asynchronous system – outputs can change state at any time the input(s) change. Difficult to design and debug.
- Synchronous system – output can change state only at a specific time in the clock cycle.
  - The clock signal is a rectangular pulse train or square wave. It is distributed to all parts of the system.
  - Positive going transition (PGT) – when clock pulse goes from 0 to 1.
  - Negative going transition (NGT) – when clock pulse goes from 1 to 0.
  - Transitions are also called edges.
  - Most digital systems are principally synchronous.
  - The speed of the synchronous system depends on clock speed.
  - A clock period is measured between PGT to the next PGT, seconds/cycle (T).
  - The speed of the system is normally referred as number of cycles in 1 second, Frequency of the clock, (Hertz = 1 cycle/second).



5-5 Clock Signals and Clocked Flip-Flops

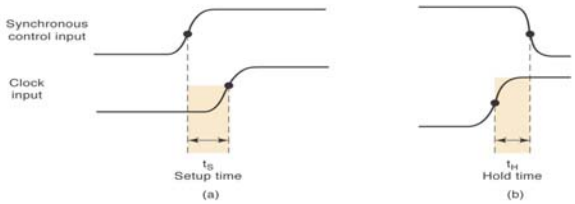
- Clocked FFs change state on one or the other clock transitions. Some common characteristics:
  - Clock inputs are labeled CLK, CK, or CP mainly edge-triggered.
  - A small triangle at the CLK input indicates that the input is activated with a PGT.
  - A bubble and a triangle indicates that the CLK input is activated with a NGT.
  - Control inputs have an effect on the output only at the active clock transition (NGT or PGT). These are also called synchronous control inputs.
  - The control inputs get the FF outputs ready to change (determine What), but the change is not triggered until the CLK edge (determine when).



5-5 Clock Signals and Clocked Flip-Flops

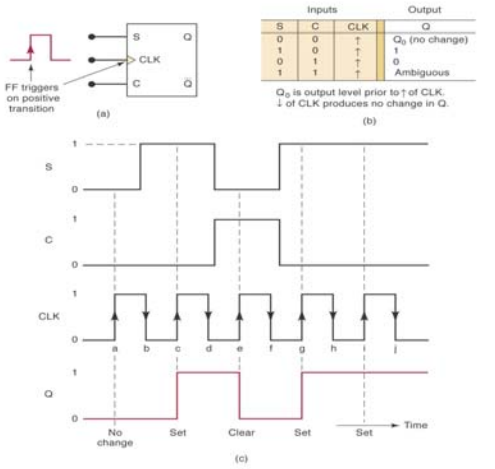
- Setup time, t_S is the minimum time interval before the active CLK transition that the control input must be kept at the proper level.
- Hold time, t_H is the time following the active transition of the CLK during which the control input must kept at the proper level.

The control inputs must be stable for at least t_S(min) prior the clk transition & at least t_H(min) after the clk transition



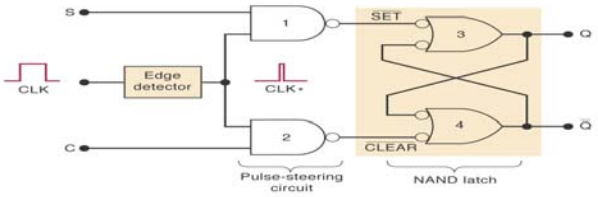
5-6 Clocked S-C Flip-Flop

- The set-clear (or set-reset) FF will change states at the positive going or negative going clock edge.
- FF is only affected by PGT transition at points (a, c, e, g, i)

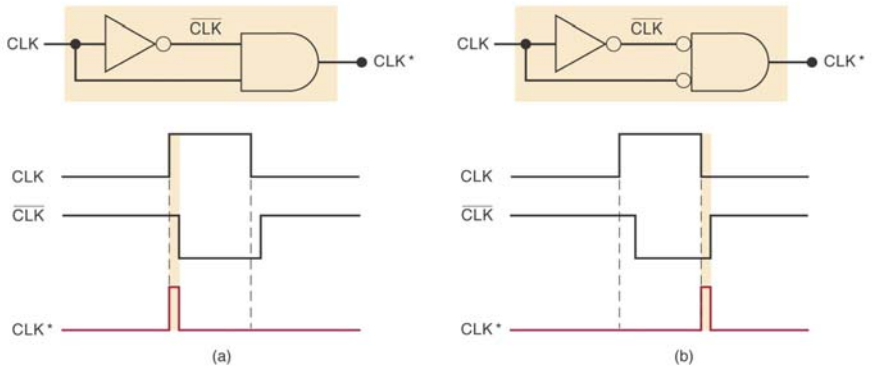


internal circuitry for an edge-triggered S-C flip-flop

- Basic NAND gate latch formed by NAND(3,4)
- Pulse-steering circuit formed by NAND(1,2)
- Edge-detector circuit
- Edge detector produces a narrow positive going spike (CLK*) that coincident with the PGT of the CLK
- The pulse circuits steers the spike through to the SET or CLEAR input in accordance with the level present on S and C
- When S=1, C=0, the CLK* produces a low pulse at the SET input of the latch.

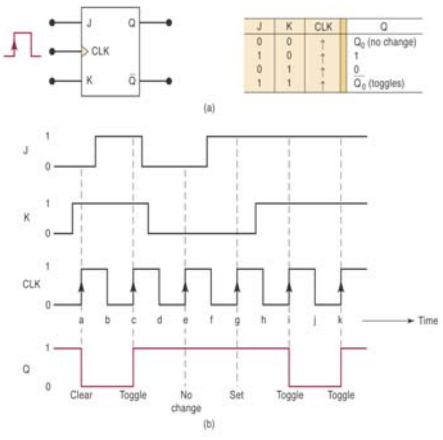


Implementation of edge-detector circuits used in edge-triggered flip-flops: (a) PGT; (b) NGT.



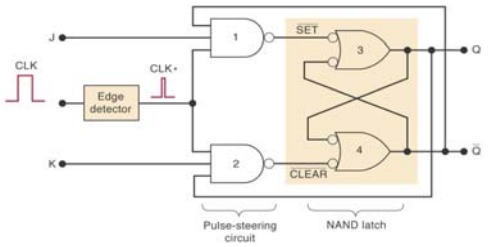
5-7 Clocked J-K Flip-Flop

- Operates like the S-C FF. J is set, K is clear.
- When J and K are both high the output is toggled from whatever state it is in to the opposite state.
- May be positive going or negative going clock trigger.
- Has the ability to do everything the S-C FF does, plus operate in toggle mode.



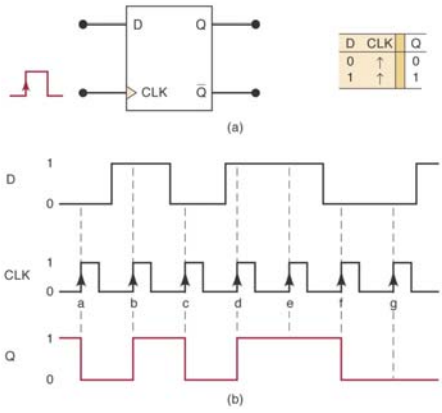
internal circuitry for an edge-triggered J-K flip-flop

- Same as edge-triggered S-C flip-flop
- The only difference is that the Q, $\overline{Q}$ outputs are fed back to the pulse-steering NAND gate, this causes J-K to toggle for J=K=1
- Assume J=K=1 and Q is low, NAND gate 1 steers CLK* to $\overline{SET}$ of the NAND latch to produce Q = 1.
- The opposite will occur if we starts with Q=1



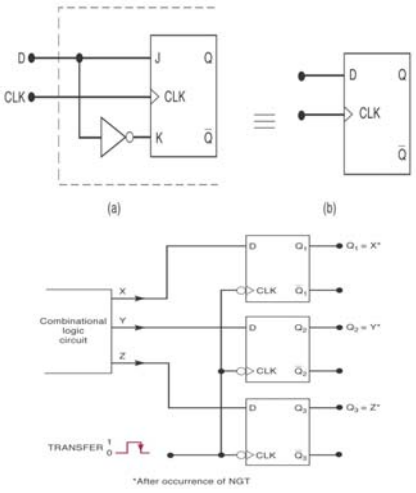
5-8 Clocked D Flip-Flop

- One data input.
- The output changes to the value of the input at either the positive going or negative going clock trigger.
- May be implemented with a J-K FF by tying the J input to the K input through an inverter.
- Useful for parallel data transfer.



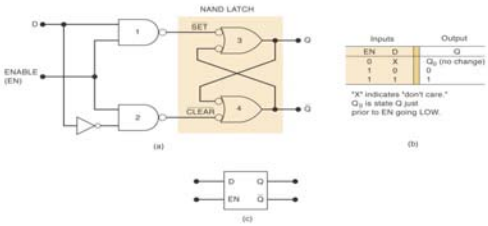
Edge-triggered D flip-flop implementation from a J-K flip-flop

- D flip-flop is implemented by adding a single inverter to the edge-triggered J-K flip-flop.
- Why using D flip-flop: Q takes the value of D input on controlled timing PGT (can be NGT too)
- Example: Outputs of combinational circuit X,Y,Z are to be transferred for storage to Q1, Q2, Q3 simultaneously for subsequent processing.



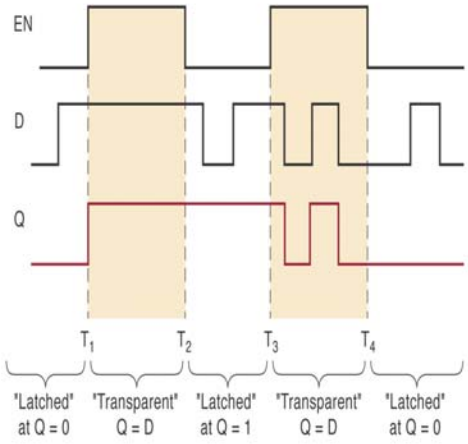
5-9 D Latch (Transparent Latch)

- One data input.
- The clock has been replaced by an enable line.
- The device is NOT edge triggered (level).
- The output follows the input only when EN is high. When EN is low, D input is inhibited from affecting the NAND latch, output of the steering gates will be held HIGH.



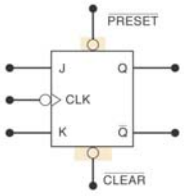
Example 5-7 showing the two modes of operation of the transparent *D* latch.

- Assume Q = 0, determine the waveform for a D latch.
- Before T1, EN is low, so Q is latched at its current o level and can not change with D.
- During T1 to T2, EN is high, Q will follow D, when EN goes low at T2, Q will latch at high level and remain high.
- At T3 when EN goes high, Q will follow D until T4 where EN goes low, Q will stay low and not follow D.



5-9 Asynchronous Inputs

- Inputs that depend on the clock are synchronous.
- Most clocked FFs have asynchronous inputs that do not depend on the clock.
- The labels PRE and CLR are used for asynchronous inputs. The asynchronous inputs are override inputs.
- Active low asynchronous inputs will have a bar over the labels and inversion bubbles.
- If the asynchronous inputs are not used they will be tied to their inactive state (Why?).

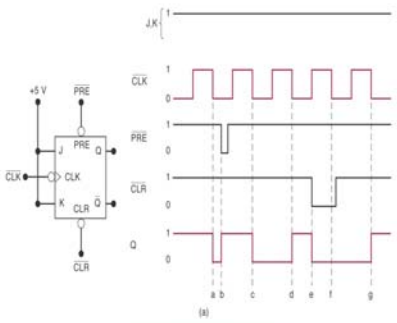


PRESET	CLEAR	FF response
1	1	Clocked operation*
0	1	Q = 1 (regardless of CLK)
1	0	Q = 0 (regardless of CLK)
0	0	Not used

*Q will respond to J, K, and CLK.

Example 5-9 showing how a clocked flip-flop responds to asynchronous inputs.

- Assume Q = 1, determine the waveform for a J-K flip-flop.
- Initially PRE* and CLR* are in their inactive state (no effect) at point a, Q will go low (toggle) since j=k=1..
- At point b, PRE* is low, this cause Q to go high, at point c, both PRE* & CLR* are inactive, CLK NGT will cause Q to toggle low and toggled back high at point d.
- At point e, CLR* is active, q will go low, the NGT of CLK at f will have no effect since CLR* is active.
- At point g, NGT of CLK will cause Q to toggle to high.



Point	Operation
a	Synchronous toggle on NGT of CLK
b	Asynchronous set on PRE = 0
c	Synchronous toggle
d	Synchronous toggle
e	Asynchronous clear on CLR = 0
f	CLR overrides the NGT of CLK
g	Synchronous toggle

(b)

5-14 Flip-Flop Applications

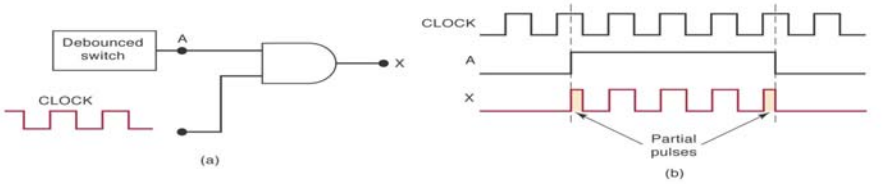
- Examples of applications:
  - Counting
  - Storing binary data
  - Transferring binary data between locations
- Many FF applications are categorized as sequential circuits, which means that the output follows a predetermined sequence of states, with a new state occurring each time a clock pulse occurs.

5-15 Flip-Flop Synchronization

- Most systems are primarily synchronous in operation, in that changes depend on the clock.
- Asynchronous and synchronous operations are often combined.
- The random nature of asynchronous inputs can result in unpredictable results.

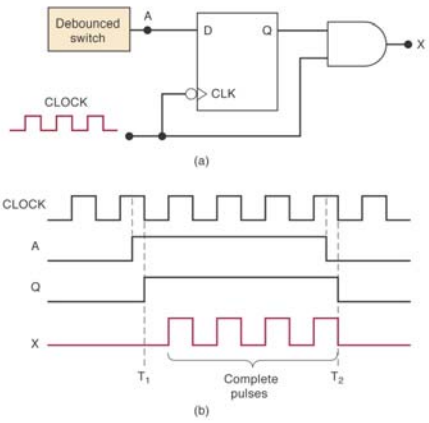
Asynchronous signal *A* can produce partial pulses at *X*.

- Most systems are primarily synchronous in operation, in that changes depend on the clock.
- Asynchronous and synchronous operations are often combined.
- The random nature of asynchronous inputs can result in unpredictable results.
- This can produce partial clock pulses at output X if either transition of A occurs while the clock signal is HIGH.
- To solve this problem is by connecting A to the D flip-flop.



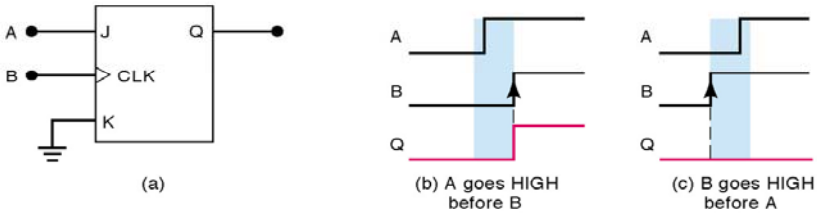
An edge-triggered D flip-flop is used to synchronize the enabling of the AND gate to the NGTs of the clock.

- When A goes HIGH, Q will not go High until the next NGT of the clock at T1.
- Now the HIGH at Q will enable the AND gate to pass subsequent complete clock pulses to X.
- When A goes LOW, Q will not go LOW until the next NGT of the clock at T2 and therefore the AND gate will not inhibit clock pulses until the clock pulses that ends at T2 has passed to X.
- This way X will contain complete clock pulses.



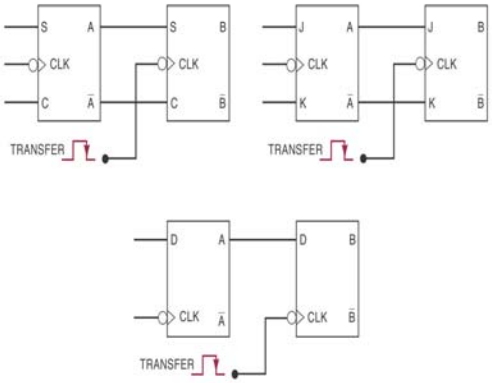
5-16 Detecting an Input Sequence

- FFs provide features that pure combinational logic gates do not.
- If an output is desired only when inputs change state in sequence, an arrangement below can be used.
- Q will go HIGH only if A goes HIGH before B goes HIGH. This is because A must be HIGH in order for Q to go HIGH on the PGT of B. This is different from AND gate only which goes high when both A and B are HIGH regardless of which input goes HIGH first.
- Can you think of timing requirements here?



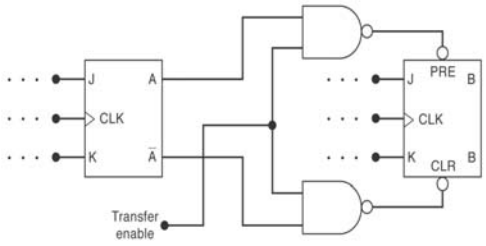
5-17 Data Storage and Transfer

- FFs are commonly used for storage and transfer of data in binary form.
- Groups of FFs used for storage are registers.
- Data transfers take place when data is moved between registers or FFs.
- The logic value stored in FF A is transferred to FF B on the NGT of the Transfer pulse
- Synchronous transfers take place at PGT or NGT of clock.
- Transfer can happen asynchronously using PRESET and CLEAR inputs.



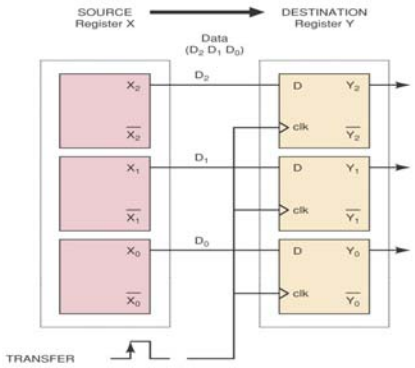
5-17 Data Storage and Transfer

- Asynchronous transfers are controlled by PRE and CLR inputs. When Transfer Enable is low, the two NAND gate are HIGH (Asynchronous inputs are inactive)
- When the Transfer Enable is HIGH, depends on the state of A, the FF either is SET or Cleared.



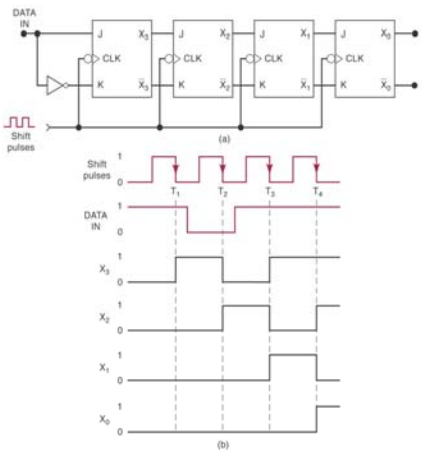
5-17 Parallel Data Transfer

- Transferring the bits of a register simultaneously is a parallel transfer.
- Register X contains 3 FFs, register Y contains 3 FFs
- On PGT of TRANSFER, the content of Register X is transferred to Register Y.
- Transferring the bits of a register a bit at a time is a serial transfer. The content of X will be transferred to Y one bit at a time



5-18 Serial Data Transfer: Shift Registers

- When FFs are arranged as a shift register, bits will shift with each clock pulse.
- JK FFs are used for 4-bit shift register, Data In is shifted into X3, X3 transfers into X2, X2 into X1 and so on, the transfer happens on NGT CLK (Shift Right).
- Initially all FFs are cleared. At T1, each X2, X1, X0 will have J=0, K=1, X3 will have J=1, K=0, only X3 will go high. At T2 X3 will have J=0, K=1, X2 will have J=1, K=0, X2 and X1 will have J=0, K=1, only X2 will go high.

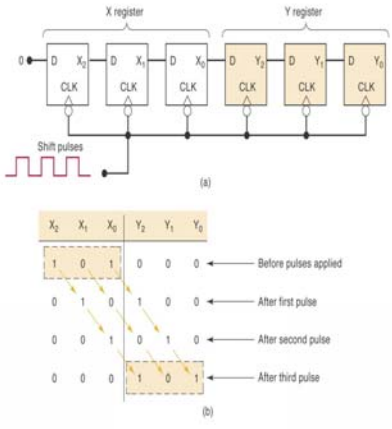


5-18 Serial Data Transfer: Shift Registers

- The direction of data shifts will depend on the circuit requirements and the design.
- The content of shift register can be serially transferred to other register

5-18 Serial Data Transfer: Serial Transfer between registers

- The content of X register is shifted into Y register.
- Each register is composed of 3 D FFs, LSB of X register is shifted into MSB of Y.
- After 3 clock pulses, the contents of Register X is shifted into Register Y and Register X will be cleared.

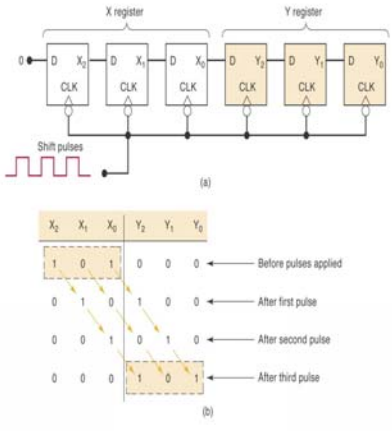


5-18 Serial Data Transfer: Shift Registers – Parallel versus Serial Transfer

- Parallel transfers – register contents are transferred simultaneously with a single clock cycle.
- Serial transfers – register contents are transferred one bit at a time, with a clock pulse for each bit.
- Serial transfers are slower, but the circuitry is simpler. Parallel transfers are faster, but circuitry is more complex.
- Serial and parallel are often combined to exploit the benefits of each.

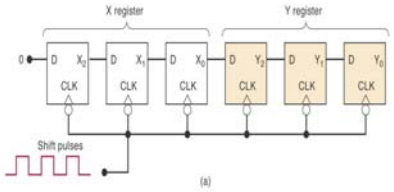
Example 5 - 13

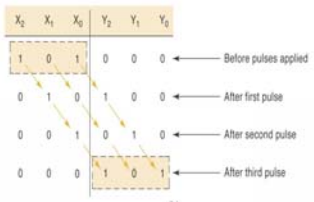
- What will be the contents of X and Y register after the occurrence of the sixth clock pulse?



Example 5 - 12

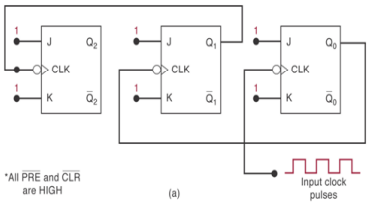
- What will be the contents of X and Y register after the occurrence of the sixth clock pulse?
- All FFs of both X and Y register will be cleared after the sixth clock pulse

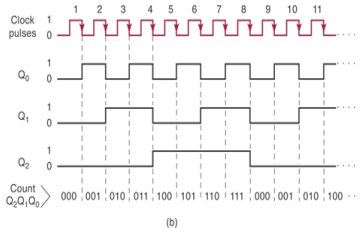




5-19 Frequency Division and Counting

- FFs are often used to divide a frequency. Each FF has J=K=1, CLK is applied only to CLK input of FF0, output of FF0 is connected to CLK of FF1, and output of FF1 is connected to CLK of FF2.
- FF0 toggles on NGT of each CLK pulse. Q0 output has a frequency ½ of CLK.
- FF1 toggles on NGT of Q0, Q1 output has a frequency ½ of Q0 and ¼ of CLK.
- What is the frequency of FF2 output?
- Using N FFs, the output frequency of the last FF is equal to $1/2^N$ of the input frequency. This is Frequency Division.





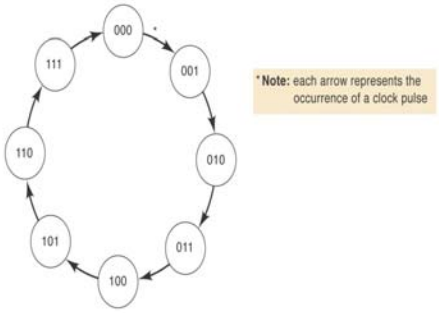
5-19 Frequency Division and Counting

- In addition to frequency divider, The same previous circuit is also acting as a binary counter by examining the sequence of the states of the FFs after each clock pulse. The outputs will count from 000_2 to 111_2 or 0_{10} to 7_{10}
- The circuit function as a binary counter in which the states of the FFs represent the number of pulses that have occurred.

2^2 Q_2	2^1 Q_1	2^0 Q_0	
0	0	0	Before applying clock pulses
0	0	1	After pulse #1
0	1	0	After pulse #2
0	1	1	After pulse #3
1	0	0	After pulse #4
1	0	1	After pulse #5
1	1	0	After pulse #6
1	1	1	After pulse #7
0	0	0	After pulse #8 recycles to 000
0	0	1	After pulse #9
0	1	0	After pulse #10
0	1	1	After pulse #11
.	.	.	. . .
.	.	.	. . .
.	.	.	. . .

5-19 Frequency Division and Counting – State Transition Diagram

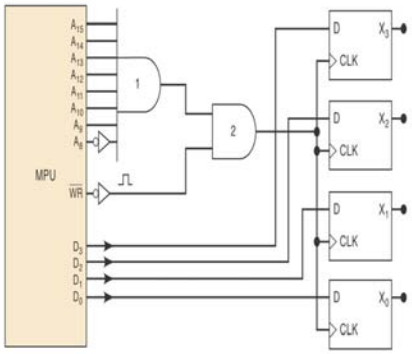
- The state transition of the FFs can be represented by State Transition Diagram.
- Each circle represents one possible state, the arrows connecting the states show how one state changes to another when a CLK pulse is applied.
- State Transition diagram is used in describing and designing sequential circuit.



- The number of states possible in a counter is the modulus or MOD number. This counter is a MOD-8 (2^3) counter. If another FF is added it would become a MOD-16 (2^4) counter. MOD-2^N

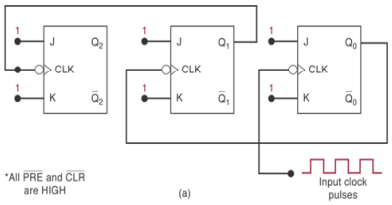
5-20 Microcomputer Application

- Microprocessor units (MPUs) which will be studied later, perform many functions that involve the use of registers for data transfer and storage.
- MPUs may send data to external registers for many purposes, including:
  - relay control
  - Motor starting
  - Device positioning
  - Motor speed controls



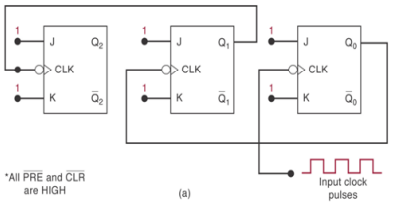
Example 5 - 14

- If the circuit of 3-bit counter is changed to have 6 FFs:
  - Determine the counter's MOD number
  - Determine the Frequency of the last FFs (Q5) when the input frequency is 1 MHz
  - What is the range of counting states for this counter?
  - Assume a starting state (000000). What will be the counter's state after 129 pulses?



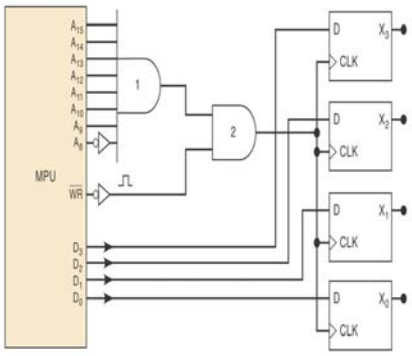
Example 5 - 14

- If the circuit of 3-bit counter is changed to have 6 FFs:
  - Determine the counter's MOD number
    - MOD number = $2^6 = 64$
  - Determine the Frequency of the last FFs (Q5) when the input frequency is 1 MHz
    - f = 1MHZ/64 = 15.625 kHz
  - What is the range of counting states for this counter?
    - 000000 to 111111 (0 to 63), number of states = 64
  - Assume a starting state (000000). What will be the counter's state after 129 pulses?
    - Counter will be back to its starting state every 64 pulses. So the 129th Pulse will bring the counter to state (000001)



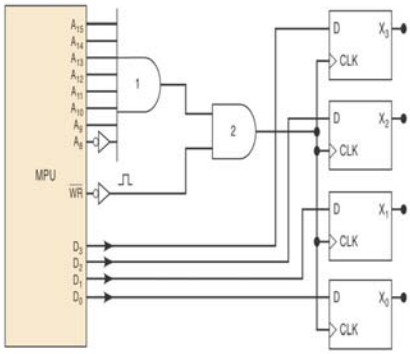
Example 5 - 15

- What address code must MPU generate in order for the data to be transferred into X?
- Assume that X3-X0 = 0110, A15-A8 = 11111111, D3-D0 = 1011. What will be the contents of X after WR* pulse occurs?



Example 5 - 15

- What address code must MPU generate in order for the data to be transferred into X?
  - $(A_{15}..A_8) = (11111110)$
- Assume that X3-X0 = 0110, A15-A8 = 11111111, D3-D0 = 1011. What will be the contents of X after WR* pulse occurs?
  - Since $A_8 = 1$, X register will not be addressed and the content will not change (0110)



Problem 1

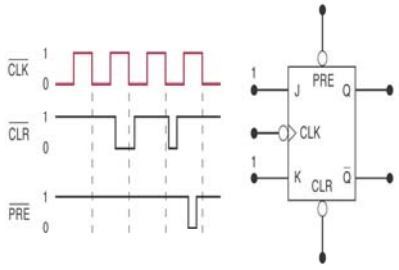
- How many FFs are required to build a binary counter to count from 0 to 1023?
  - Determine the frequency at the output of the last FF, if the input clock is 2 MHz
  - What is the counter's MOD number?
  - If the counter is initially at zero, what count will it hold after 2060 pulses?

Problem 1

- How many FFs are required to build a binary counter to count from 0 to 1023?
  - Determine the frequency at the output of the last FF, if the input clock is 2 MHz
  - What is the counter's MOD number?
  - If the counter is initially at zero, what count will it hold after 2060 pulses?
- $2^N = 1024$, N = 10 FFs
- Frequency = 2MHz/1024 = 1953 Hz
- MOD number = $2^N = 1024$
- Every 1024 pulses, counter recycles through zero, after 2060, counter will be at count 12.

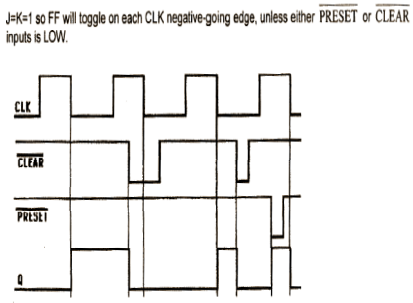
Problem - 2

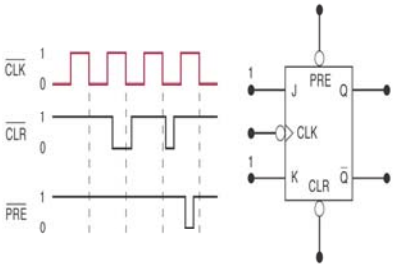
- Determine the Q waveform for the J-K FF. Assume that Q = 0 initially. (Hint: the asynchronous inputs override all other inputs) :



Problem - 2

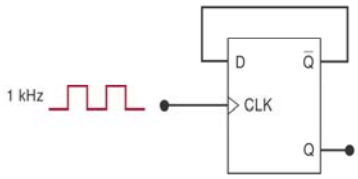
- Determine the Q waveform for the J-K FF. Assume that Q = 0 initially. (Hint: the asynchronous inputs override all other inputs) :





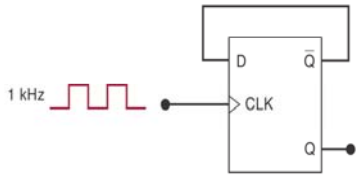
Problem 3

- An edge-triggered D flip-flop can be made to operate in the toggle mode by connecting it as shown.
  - Assume that Q = 0 initially, determine the Q waveform
  - Change the circuit so that Q is connected back to D, then determine the Q waveform.



Problem 3

- An edge-triggered D flip-flop can be made to operate in the toggle mode by connecting it as shown.
  - Assume that Q = 0 initially, determine the Q waveform
  - Change the circuit so that Q is connected back to D, then determine the Q waveform.



  - If Q initially = 0, then Q is ½ frequency of the clock, 500 Hz
  - If Q is connected to D, then Q will not change as clock applied.