CO20-320241

Computer Architecture and Programming Languages

CAPL

Lecture 7 & 8

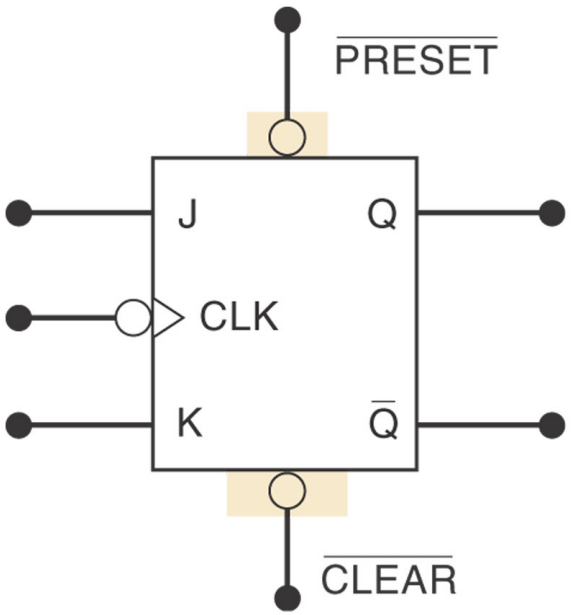
Dr. Kinga Lipskoch

Fall 2019

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
- Active low asynchronous inputs usually have a bar over the labels and inversion bubbles
- If the asynchronous inputs are not used they are usually tied to their inactive state (i.e., where no change occurs)

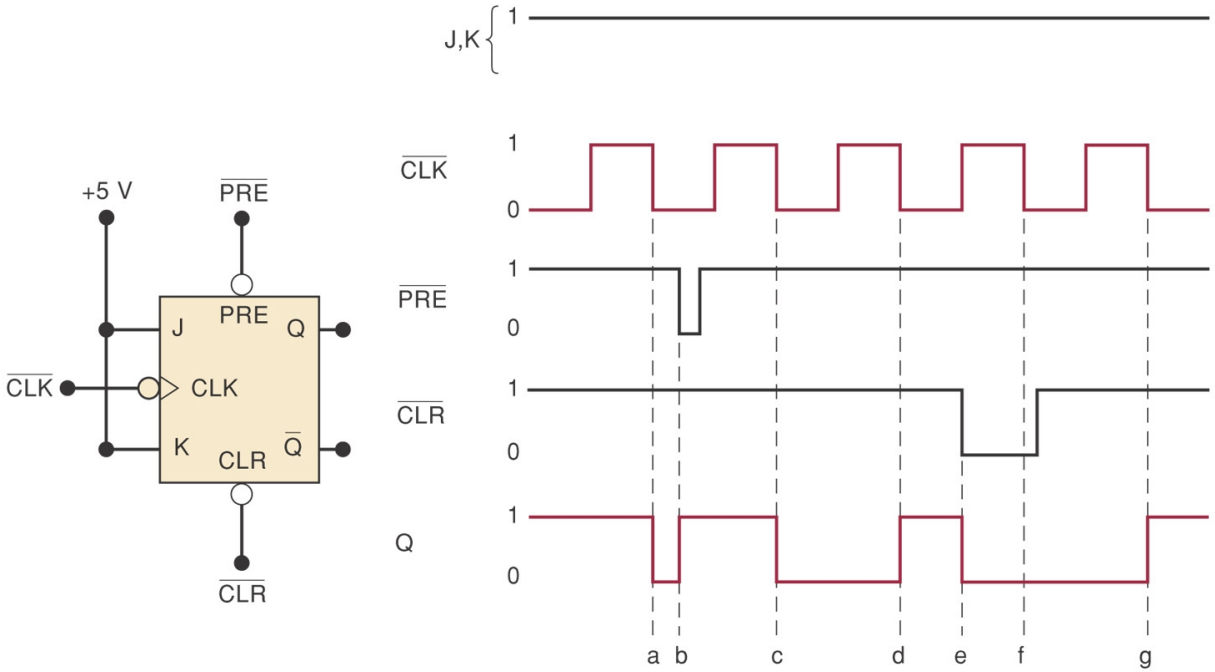
Clocked J-K Flip-Flop with Asynchronous Inputs



J	K	Clk	$\overline{PRE}$	$\overline{CLR}$	Q
0	0	↓	1	1	Q (no change)
0	1	↓	1	1	0 (Synch reset)
1	0	↓	1	1	1 (Synch set)
1	1	↓	1	1	$\overline{Q}$ (Synch toggle)
x	x	x	1	1	Q (no change)
x	x	x	1	0	0 (asynch clear)
x	x	x	0	1	1 (asynch preset)
x	x	x	0	0	(Invalid)

Waveform Example Asynchronous J-K Flip-Flop

A clocked flip-flop responding to asynchronous inputs

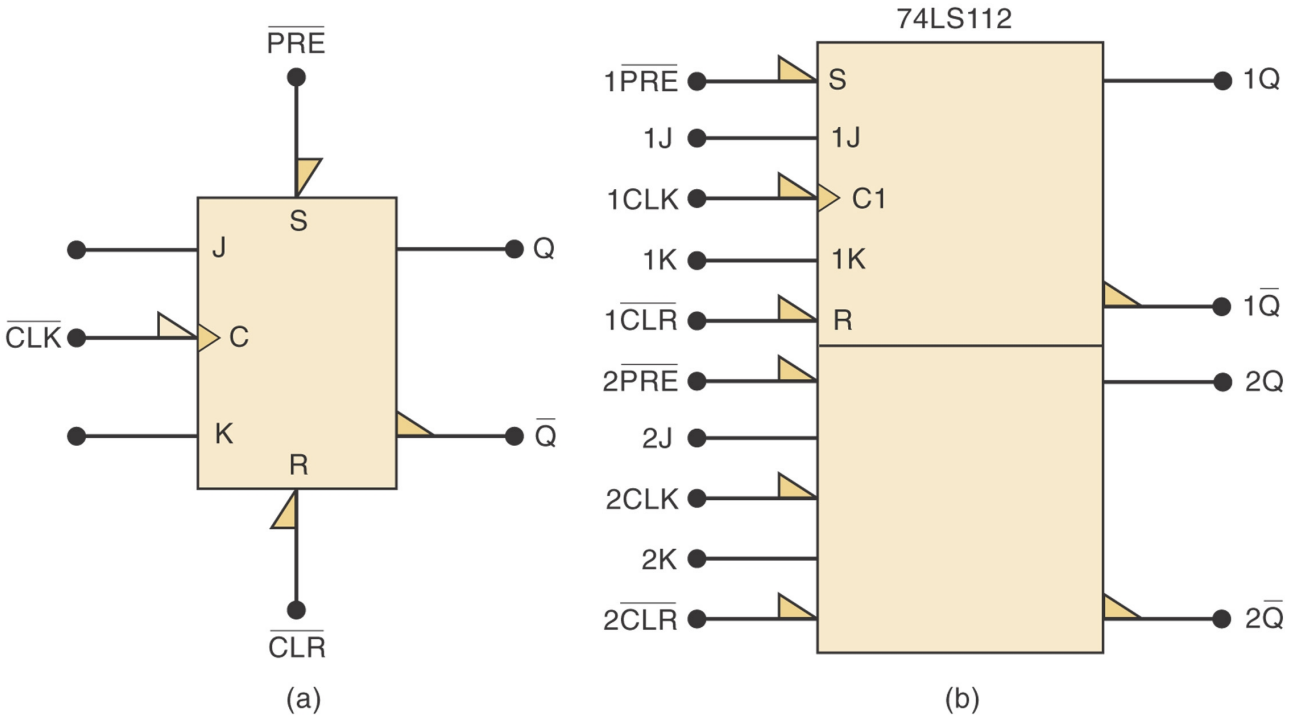


(a)

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

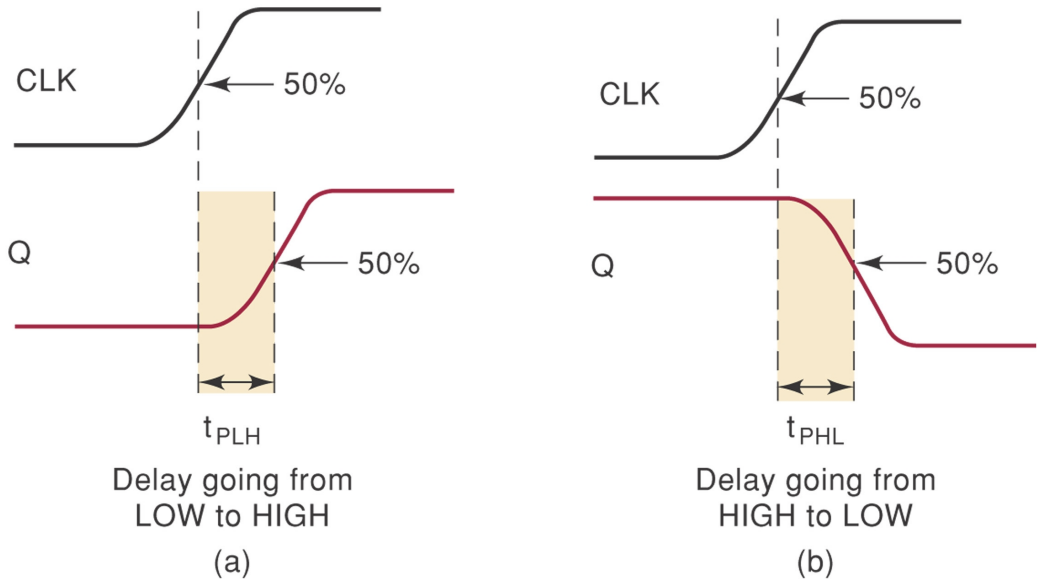
(b)

ANSI Symbols for Asynchronous J-K Flip-Flops



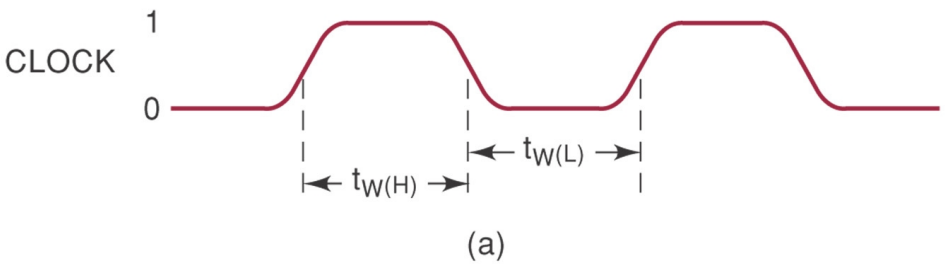
(a) a single edge-triggered J-K flip-flop and
(b) an actual IC (74LS112 dual negative-edge-triggered J-K flip-flop)

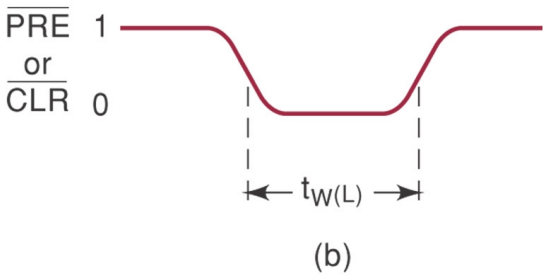
FF Propagation Delays



- Delays are measured between 50% values of input and output waveforms
- Typical delays range from a few ns to 100 ns

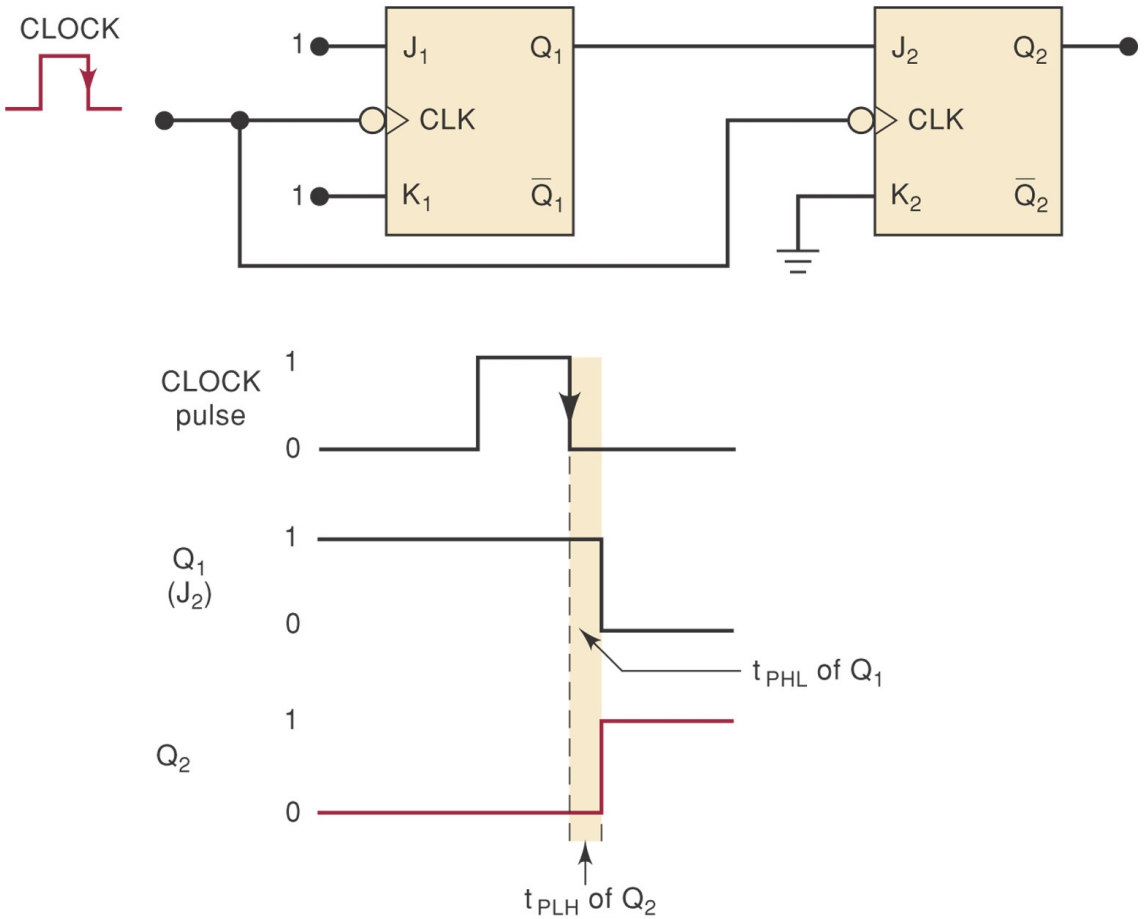
Periodic and Aperiodic Signals





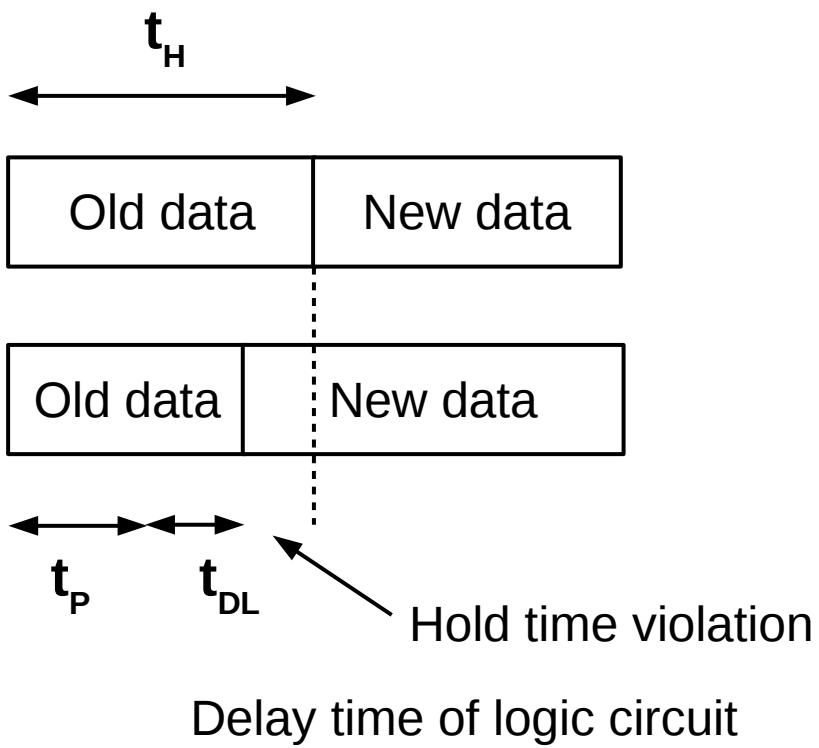
(a) Clock LOW and HIGH times
(b) Asynchronous pulse width

Timing Issues



Q_2 will respond properly to the level at Q_1 prior to the NGT of CLK, provided that J_2s hold time requirement, t_H, is less than Q_1s propagation delay

Hold Time Violation



Flip-Flop Timing Considerations (1)

Important timing parameters:

- Setup and hold times
- Propagation delay – the time for a signal at the input to be shown at the output
- Maximum clocking frequency – highest clock frequency that will give a reliable output
- Clock pulse high and low times – minimum time that clock must be high before going low, and must be low before going high

Flip-Flop Timing Considerations (2)

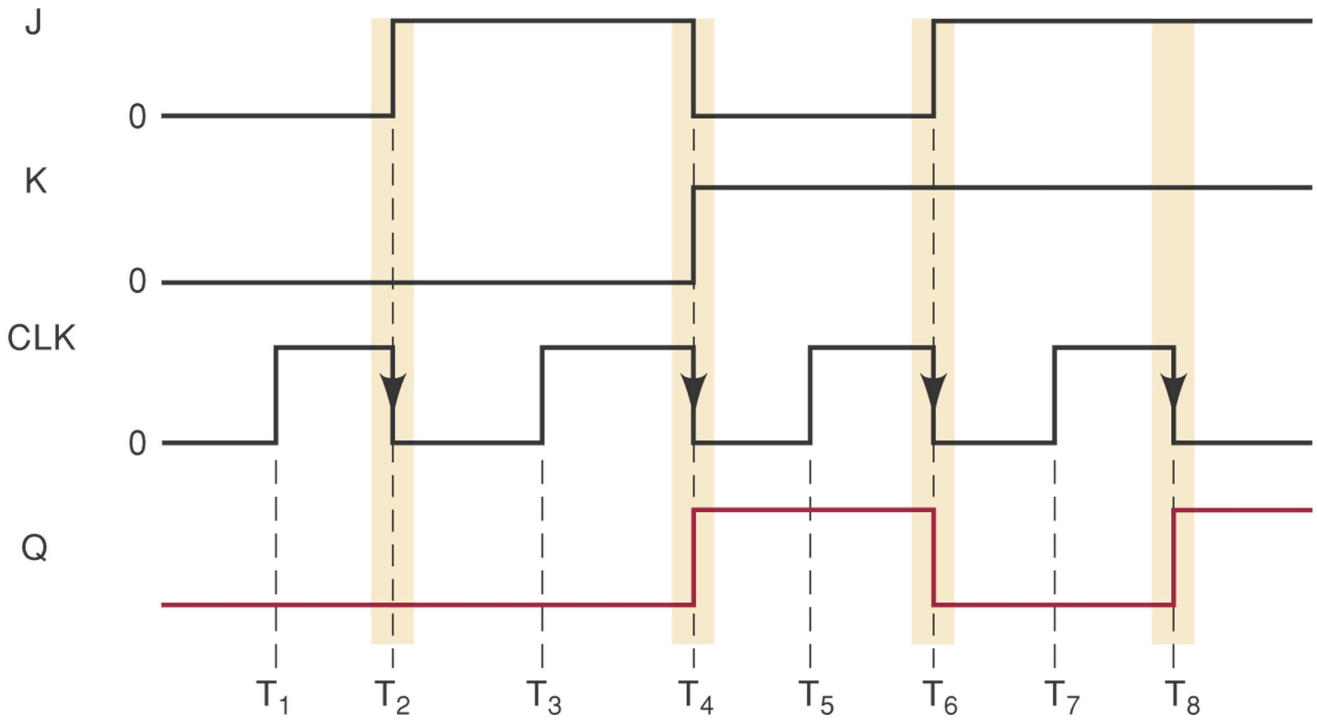
Further important timing parameters:

- Asynchronous active pulse width – the minimum time PRESET or CLEAR must be held for the FF to set or clear reliably
- Clock transition times – maximum time for the clock transitions, generally less than 50 ns for TTL (Transistor-Transistor-Logic) or 200 ns for CMOS (Complementary Metal-Oxide Semiconductor) devices

Potential Timing Problems in FF Circuits

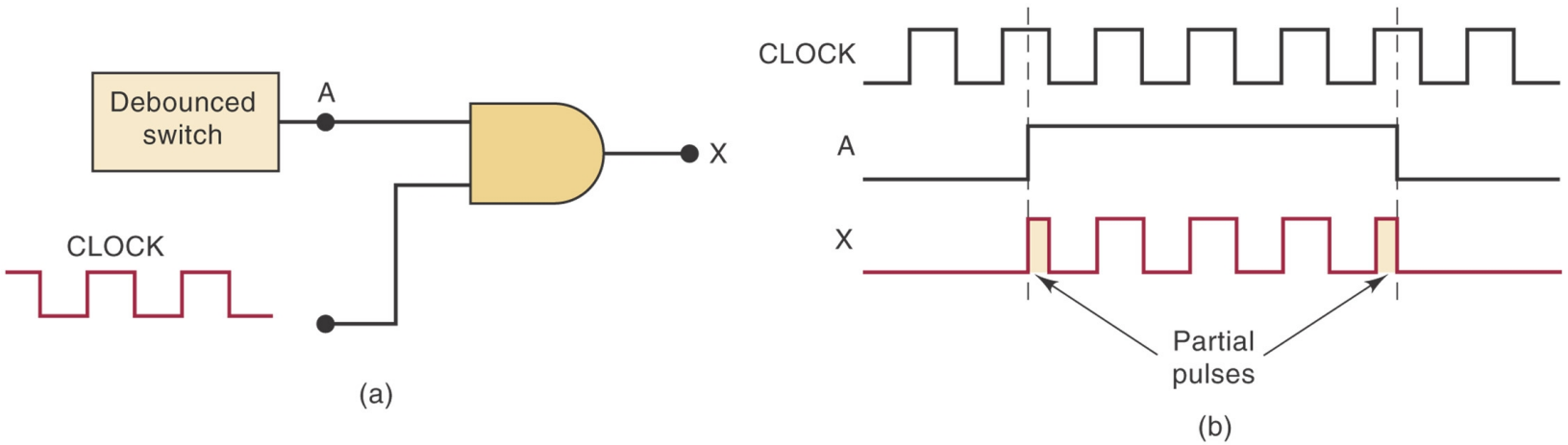
- When the output of one FF is connected to the input of another FF and both devices are triggered by the same clock, there is a potential timing problem
- Propagation delay may cause unpredictable outputs
- But, since the hold time parameter of most FFs is low, this is normally not a problem

Example for Timing Problems

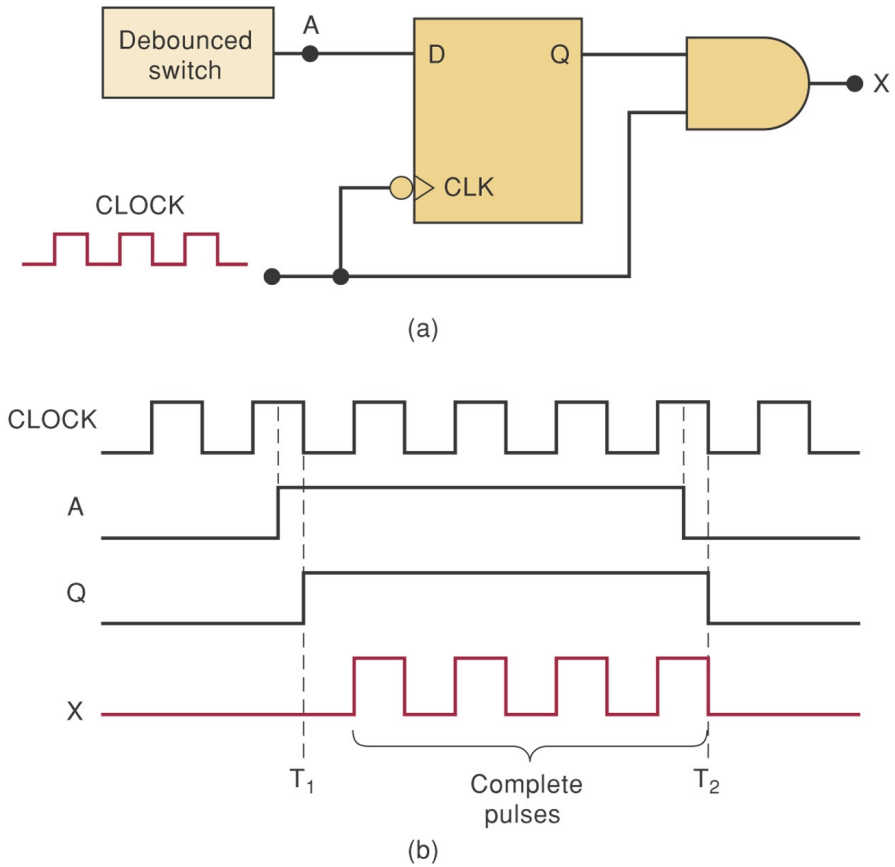


Assume $t_H = 0$ and $Q = 0$ initially

Example for Activating/Deactivating a Clock Signal (1)

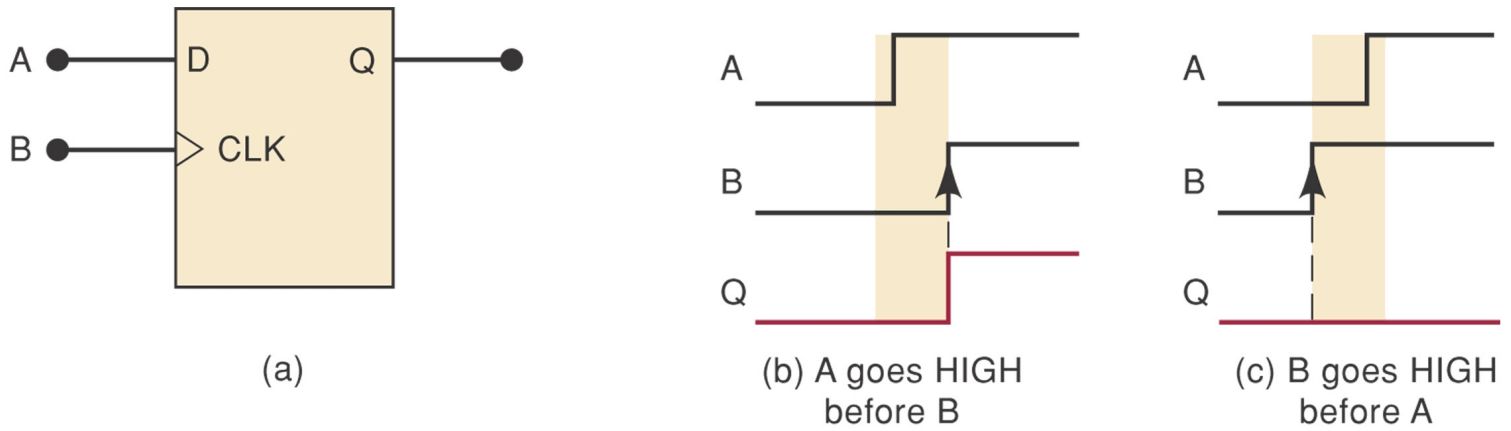


Example for Activating/Deactivating a Clock Signal (2)



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

Example for Triggering in the Order of a Given Input Sequence



Usage of Flip-Flops

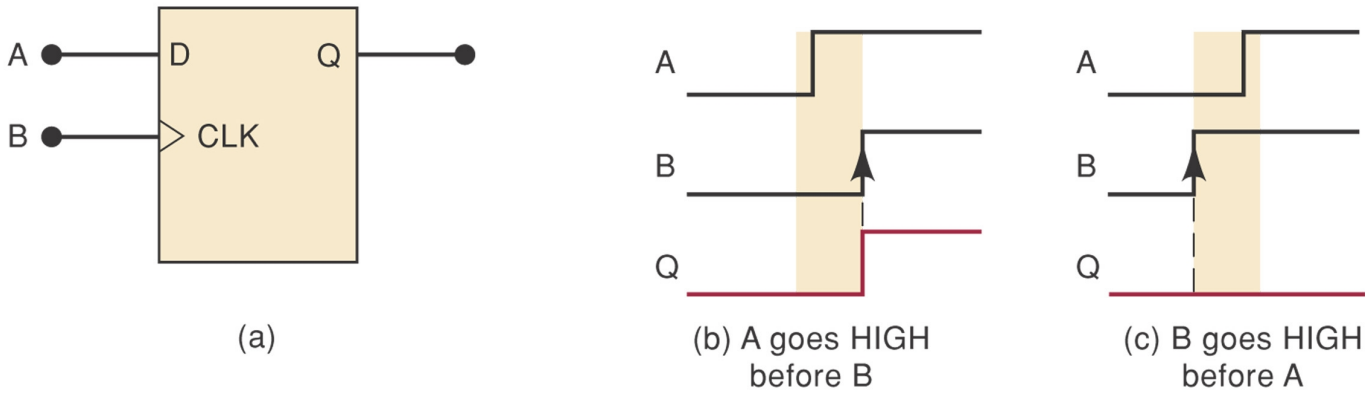
- Examples of applications:
  - Counting
  - Storing binary data
  - Transferring binary data between locations
- Many FF applications are categorized as sequential, which means that the output follows a predetermined sequence of states

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
- Example from slide 14 describes the problem and a solution

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 similar to the figure below can be used



(a)

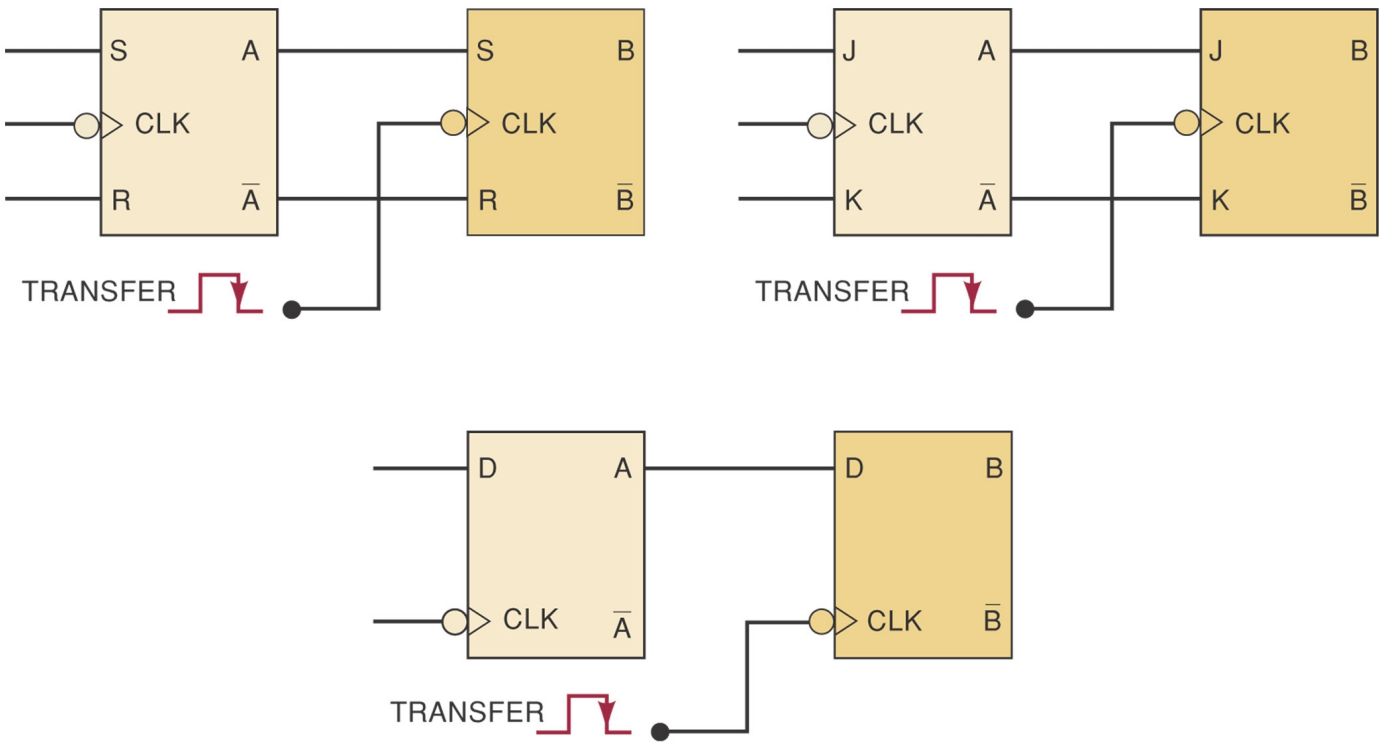
(b) A goes HIGH before B

(c) B goes HIGH before A

Data Storage and Transfer (1)

- FFs are commonly used for storage and transfer of data in binary form
- Groups of FFs used for storage are called registers
- Data transfers take place when data is moved between registers or FFs
- Synchronous transfers take place at PGT or NGT of the clock

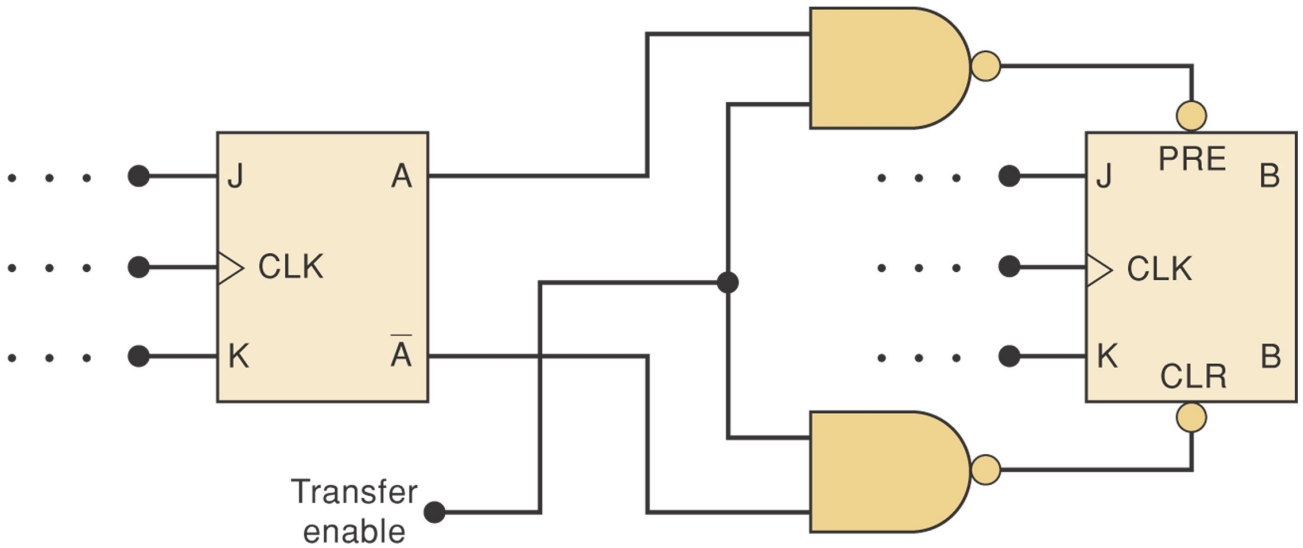
Synchronous Data Storage and Transfer



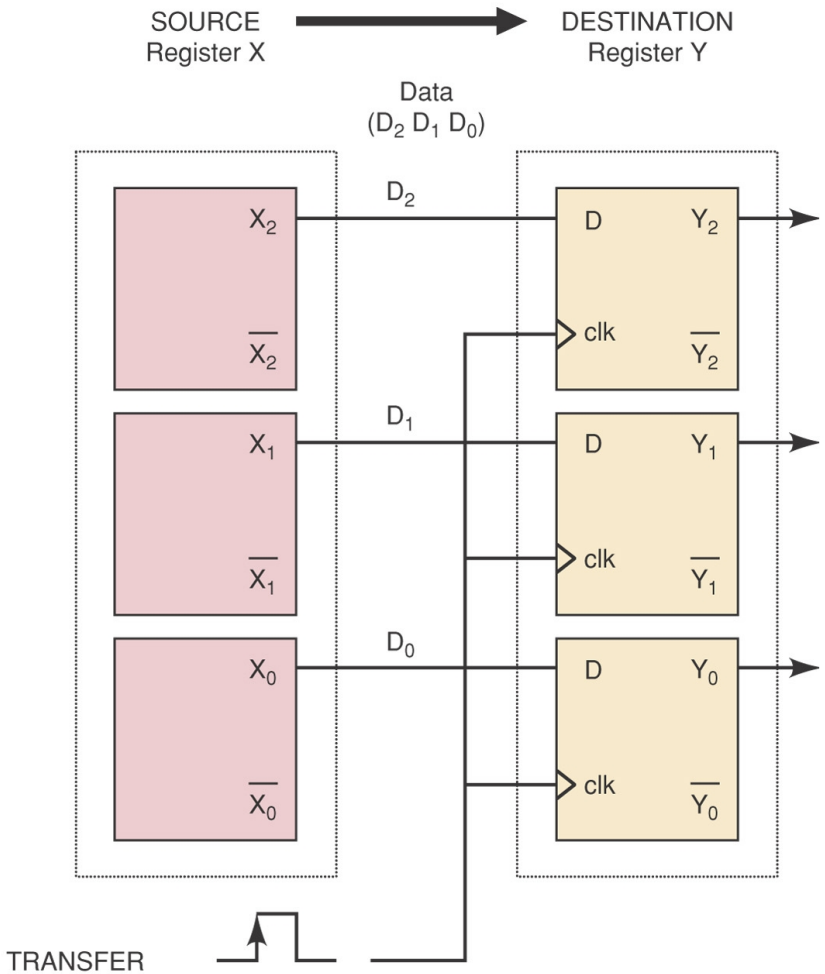
Data Storage and Transfer (2)

- Asynchronous transfers are controlled by PRE and CLR inputs
- Transferring the bits of a register simultaneously is a parallel transfer
- Transferring the bits of a register a bit at a time is a serial transfer

Asynchronous Data Storage and Transfer



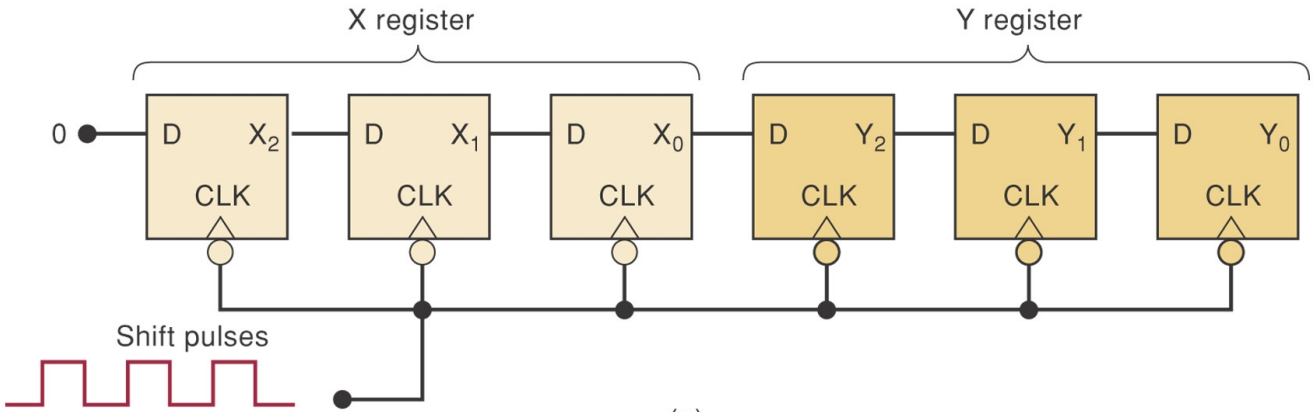
Parallel Data Storage and Transfer



Serial Data Transfer: Shift Registers (1)

- When FFs are arranged as a shift register, bits will shift with each clock pulse
- FFs used as shift registers must have very low hold time parameters to perform predictably
- Modern FFs have t_H values well within what is required
- The direction of data shifts will depend on the circuit requirements and the design

Serial Transfer of Information from Register X into Register Y

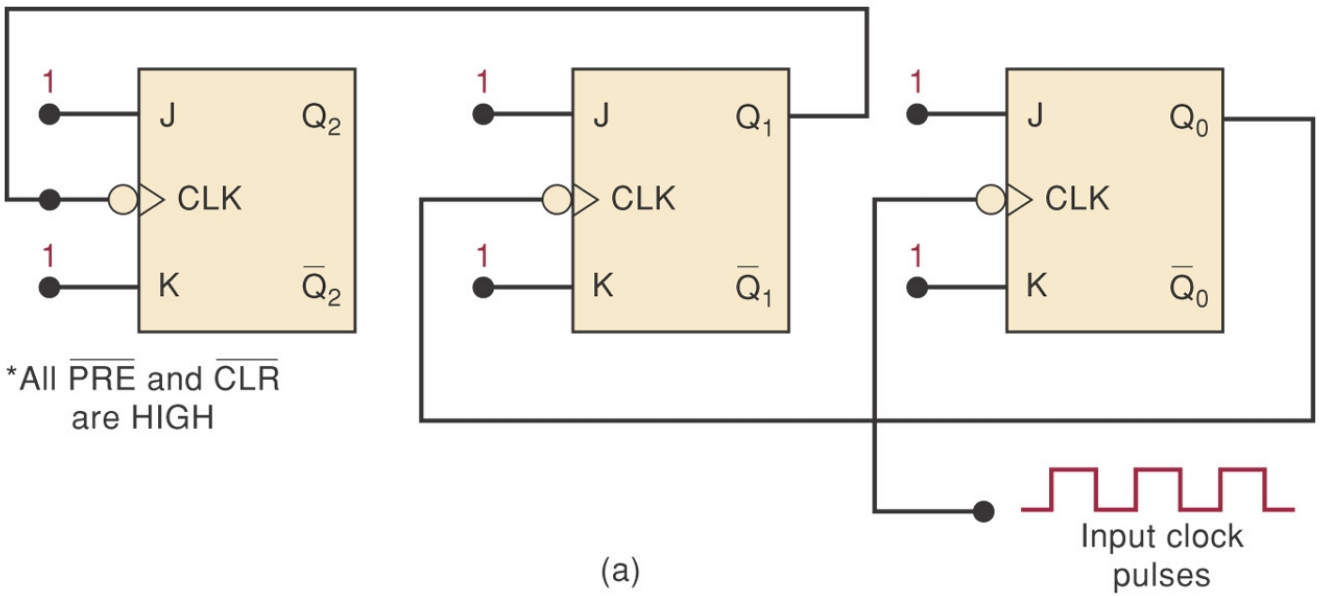


(a)

X_2	X_1	X_0	Y_2	Y_1	Y_0	
1	0	1	0	0	0	← Before pulses applied
0	1	0	1	0	0	← After first pulse
0	0	1	0	1	0	← After second pulse
0	0	0	1	0	1	← After third pulse

(b)

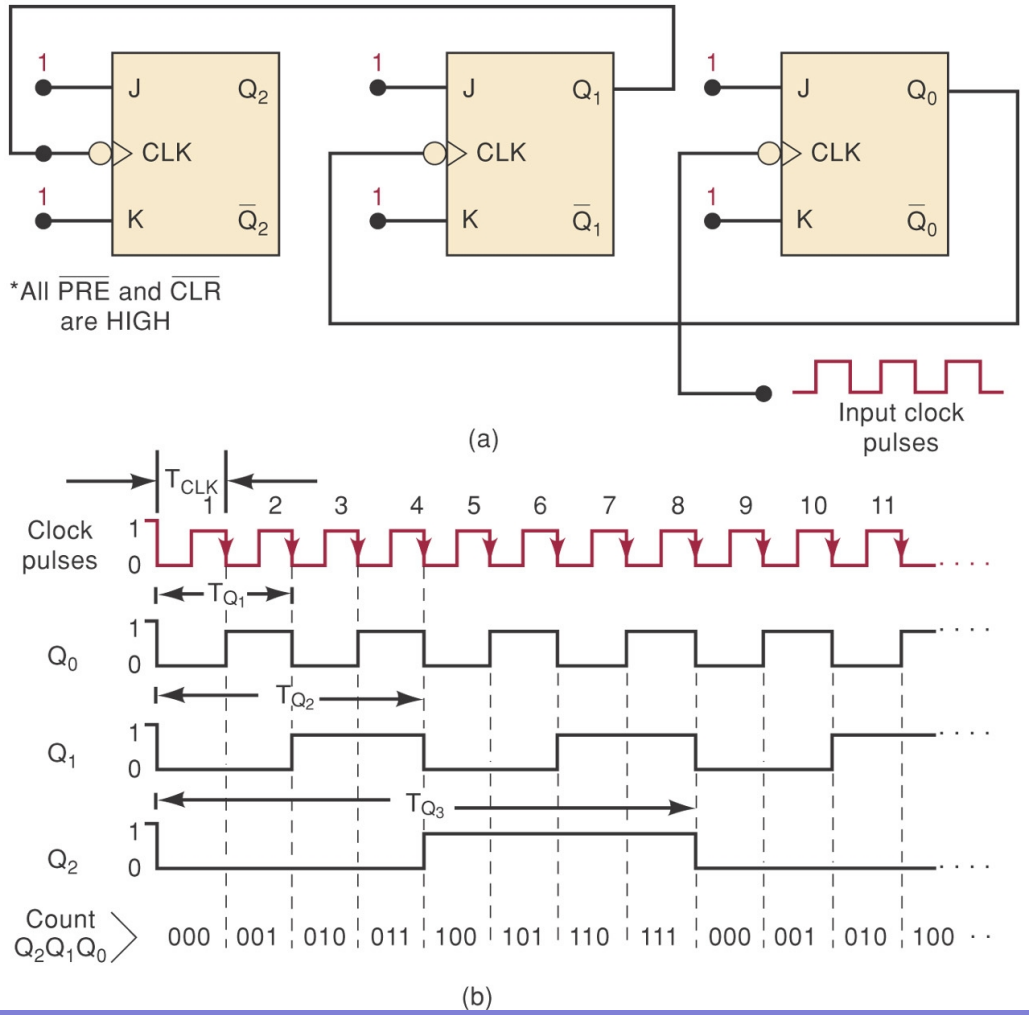
What Does this Circuit Do?



Frequency Division and Counting

- FFs are often used to divide a frequency as illustrated in figure below
- Here the output frequency is $1/8^{th}$ of the input (clock) frequency
- The same circuit is also acting as a binary counter
- The outputs will count from 000_2 to 111_2 or correspondingly 0_{10} to 7_{10}
- The number of states possible in a counter is the modulus or MOD number
- Next figure is a MOD-8 (2^3) counter, if another FF is added it would become a MOD-16 (2^4) counter

J-K Flip-Flops Wired as a Three-bit Binary Counter (MOD-8)

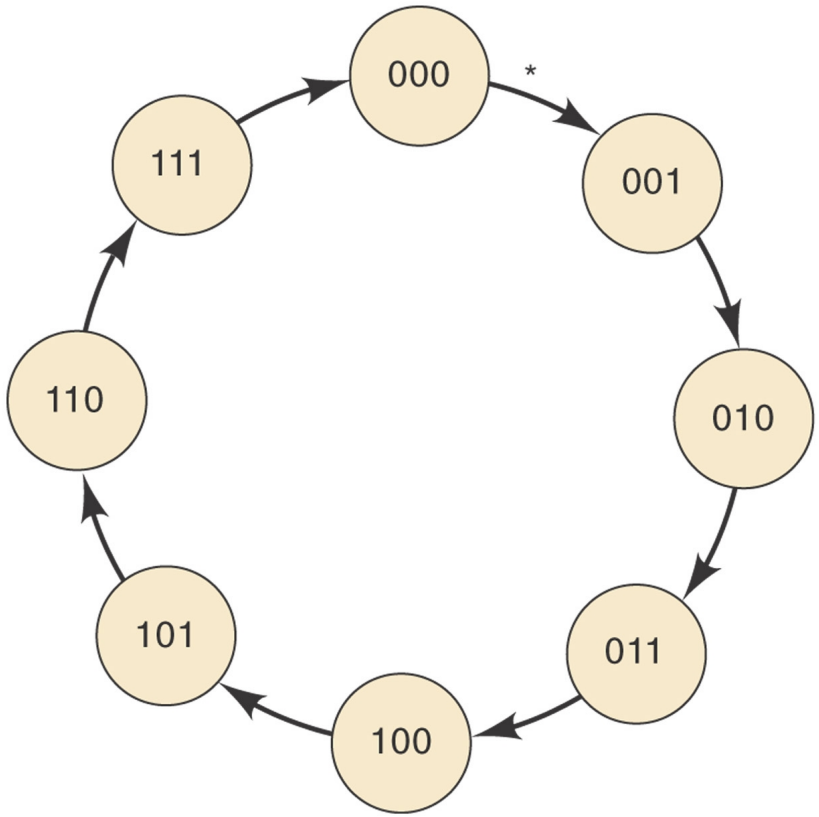


State Table

Table of flip-flop states shows a binary counting sequence

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
.	.	.	. . .
.	.	.	. . .
.	.	.	. . .

State Transition Diagram



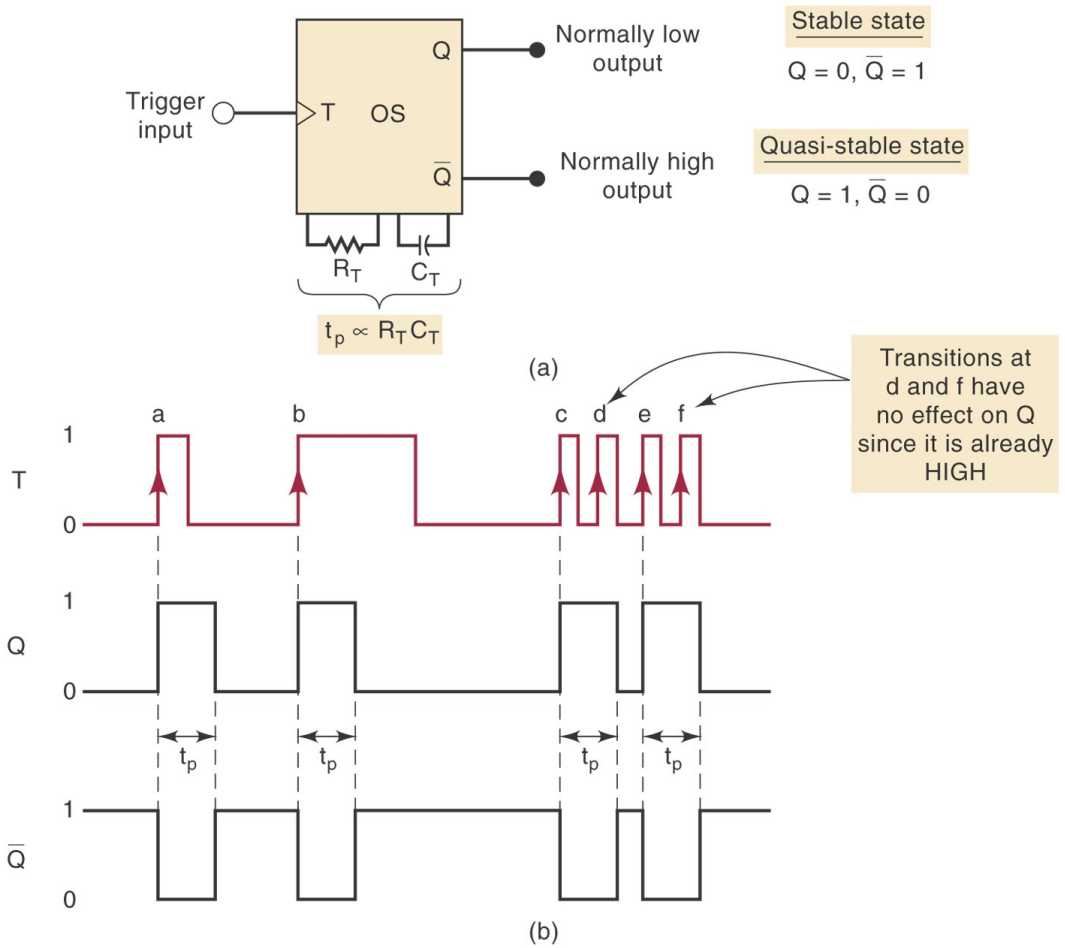
* **Note:** each arrow represents the occurrence of a clock pulse

One-shot (Monostable Multivibrator)

- Changes from stable state to quasi-stable state for a period of time determined by external components (usually resistors and capacitors)
- Non-retriggerable devices will trigger and return to stable state and after that they can be triggered again
- Retriggerable devices can be triggered while in the quasi-stable state to begin another pulse
- One shots are called monostable multivibrators because they have only one stable state
- They are prone to triggering by noise, therefore they tend to be used only in simple timing applications

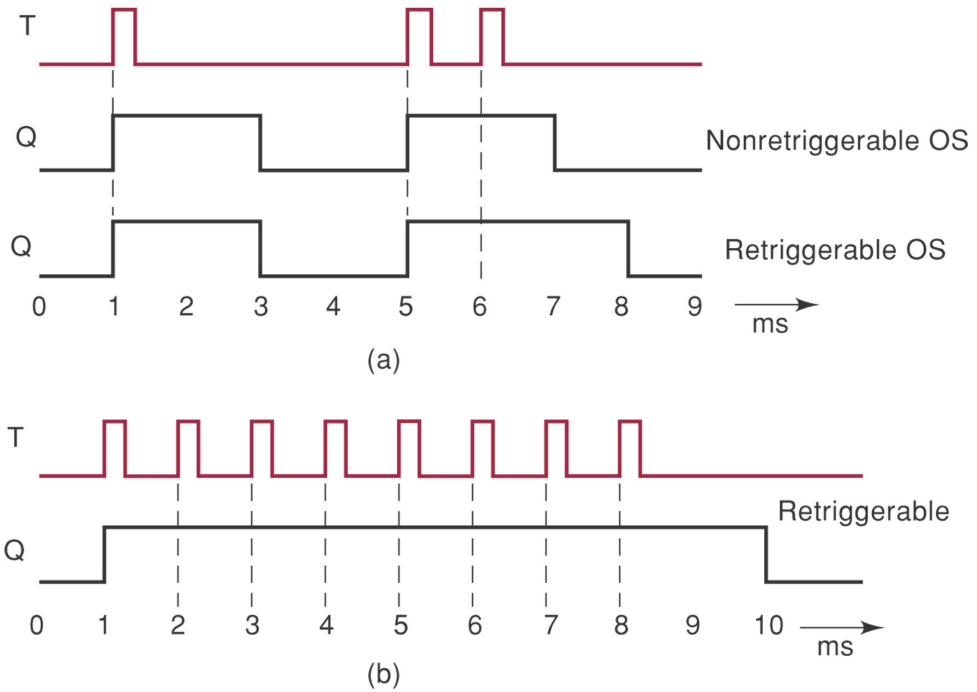
One-shot Example

- Once triggers OS outputs switch to quasi-stable state for fixed period of time t_p
- After time t_p output returns to stable state

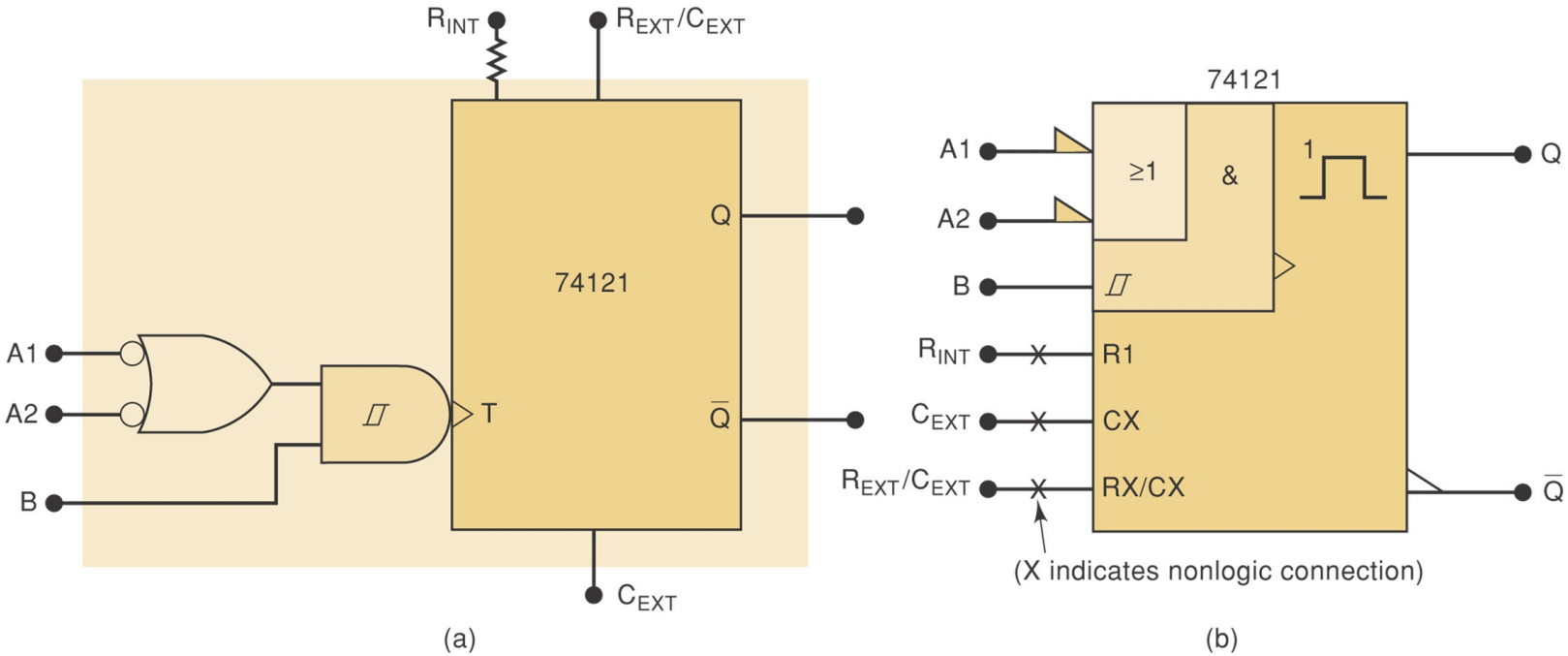


Non-retriggerable and Retriggerable One-Shots

- Retriggerable OS begins new t_p interval each time it receives a trigger pulse
- If rate of trigger pulses is fast enough, OS stays high continuously



Actual Device



Note the 1 in front of the pulse: indicates that the device is non-retriggerable

Clock Generator Circuits

- FFs have two stable states, therefore they are considered bistable multivibrators
- One-shots have one stable state and are considered monostable multivibrators
- Astable or free-running multivibrators switch back and forth between two unstable states
- This makes it useful for generating clock signals for synchronous circuits
- Crystal control may be used if a very stable clock is needed
- Crystal control is used in microprocessor based systems and microcomputers where accurate timing intervals are essential

Troubleshooting Flip-Flop Circuits

- Timing problems create some faults and symptoms that are not seen in combinational logic circuits
- Open inputs that can trigger a change in state may pick up noise resulting in erratic output
- Clock skew occurs when CLK signals arrive at different FFs at different times
- The fault may be seen only intermittently, or may disappear during testing

Digital Arithmetic: Operations and Circuits – Binary Addition –

- Binary numbers are added like decimal numbers
- In decimal, when numbers sum more than 9 a carry results
- In binary when numbers sum more than 1 a carry takes place
- Addition is the basic arithmetic operation used by digital devices to perform subtraction, multiplication, and division

Possible Representations

Sign Magnitude	One's Complement	Two's Complement
000 = +0	000 = +0	000 = +0
001 = +1	001 = +1	001 = +1
010 = +2	010 = +2	010 = +2
011 = +3	011 = +3	011 = +3
100 = -0	100 = -3	100 = -4
101 = -1	101 = -2	101 = -3
110 = -2	110 = -1	110 = -2
111 = -3	111 = -0	111 = -1

Issues: balance, number of zeros, ease of operations

Representing Signed Numbers (1)

- In order to change a binary number to 2's complement it must first be changed to 1's complement
  - To convert to 1's complement, simply change each bit to its complement (opposite)
  - To convert 1's complement to 2's complement add 1 to the 1's complement
- A positive number is true binary with 0 in the sign bit
- A negative number is in 2's complement form with 1 in the sign bit

Two's Complement Operations

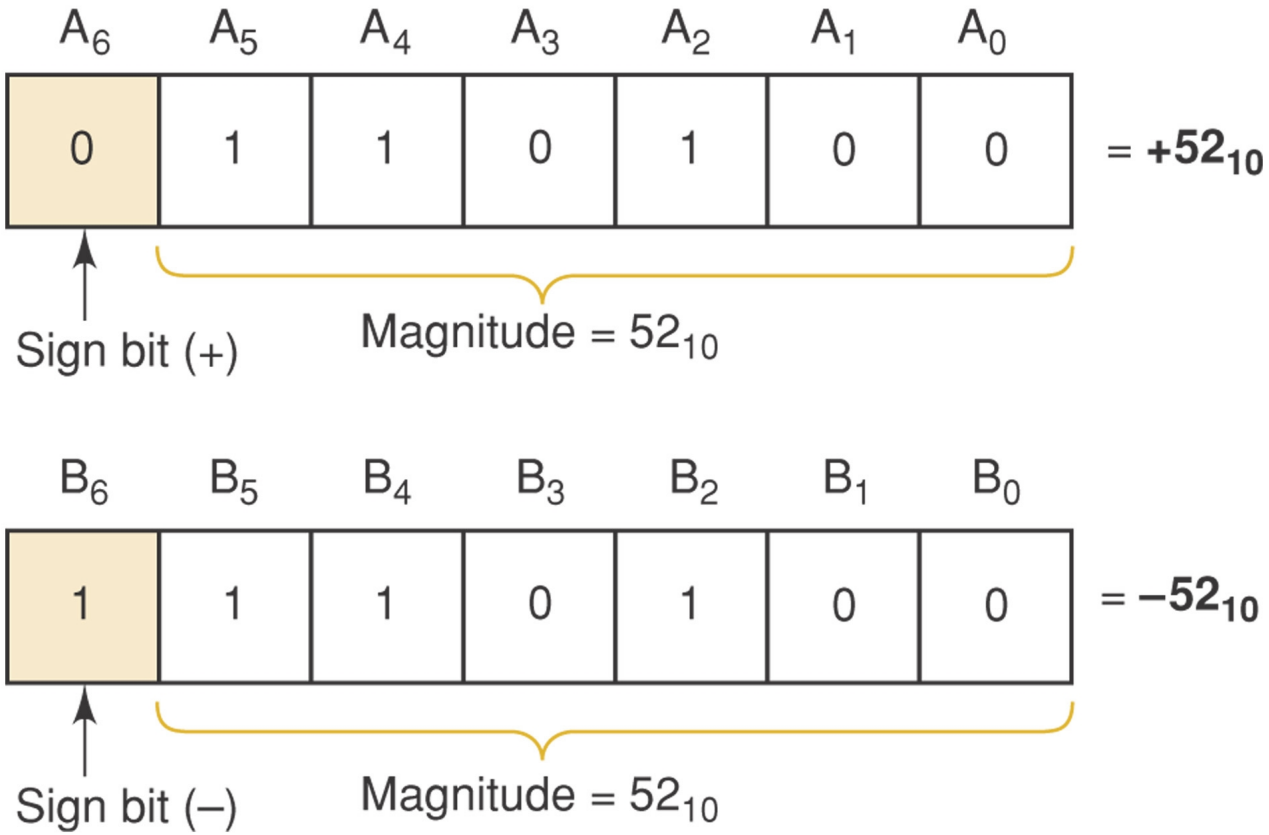
- Negating a two's complement number:
  invert all bits and add 1
  - Remember: "negate" and "invert" are different
- Example:

0000 0010		=	2
1111 1101	inverted		
1111 1110	1 added	=	-2
0000 0001	inverted		
0000 0010	1 added	=	2

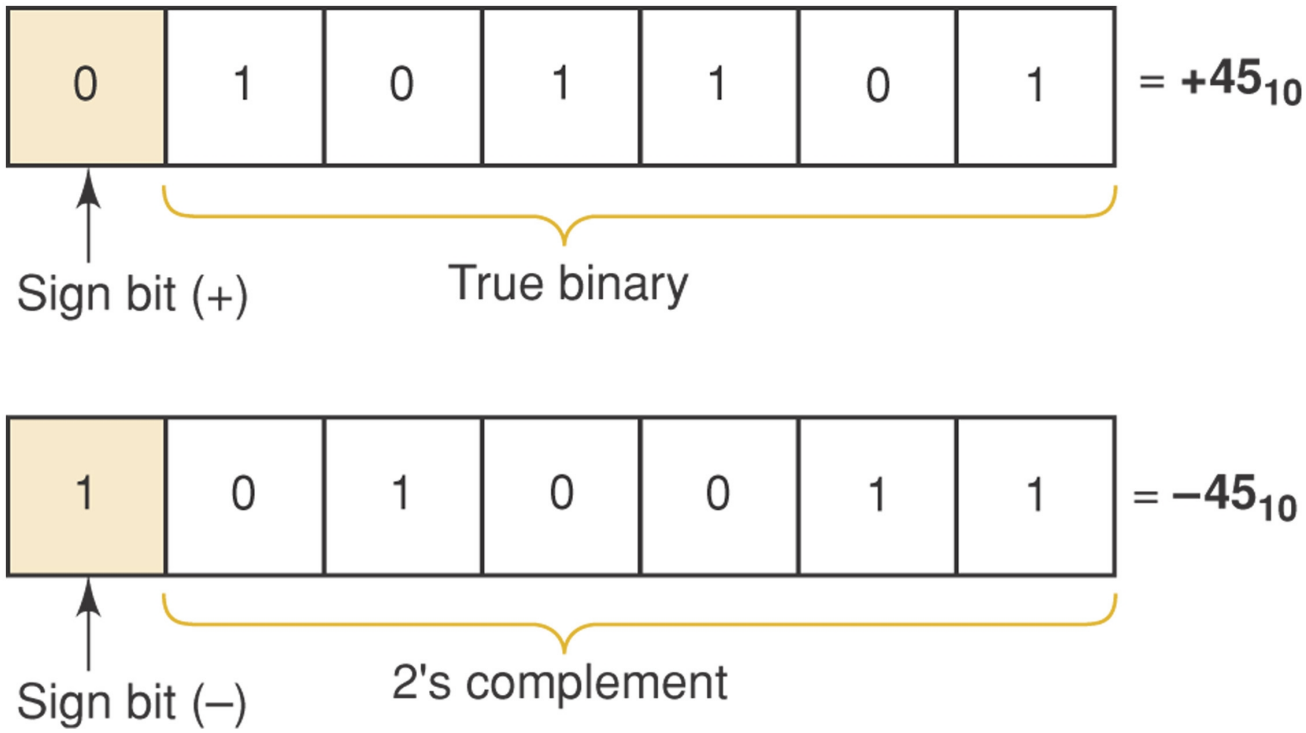
Representing Signed Numbers (2)

- A number is negated when converted to the opposite sign
- A binary number can be negated by taking the 2's complement of it

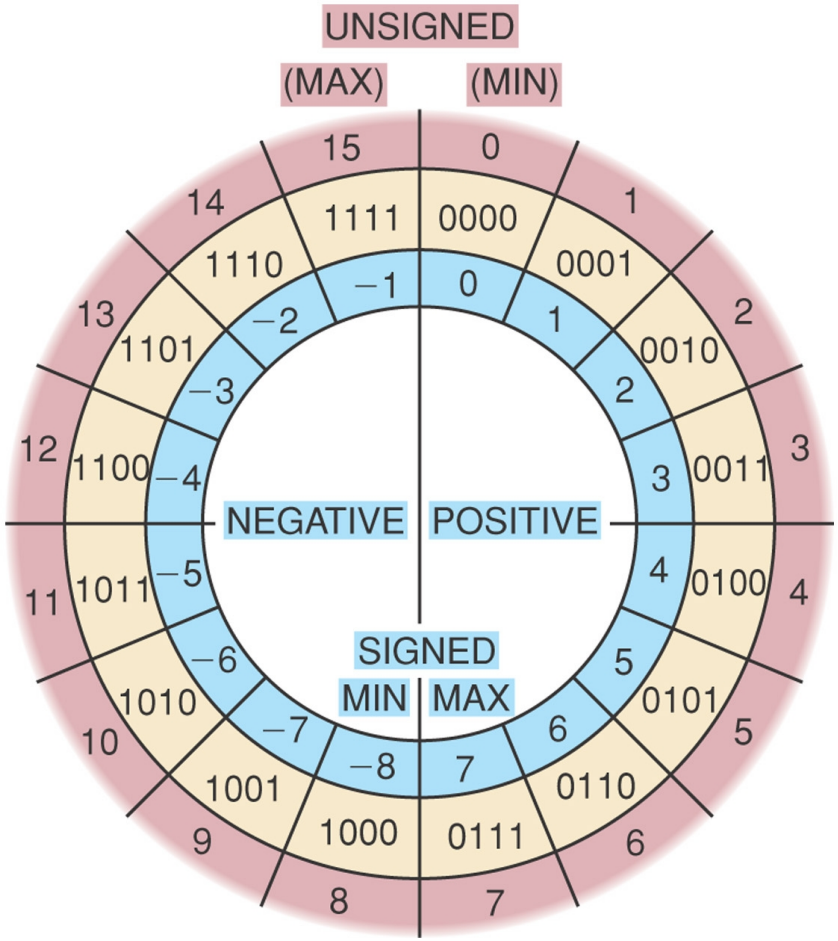
Representation of Signed Numbers in Sign-magnitude Form



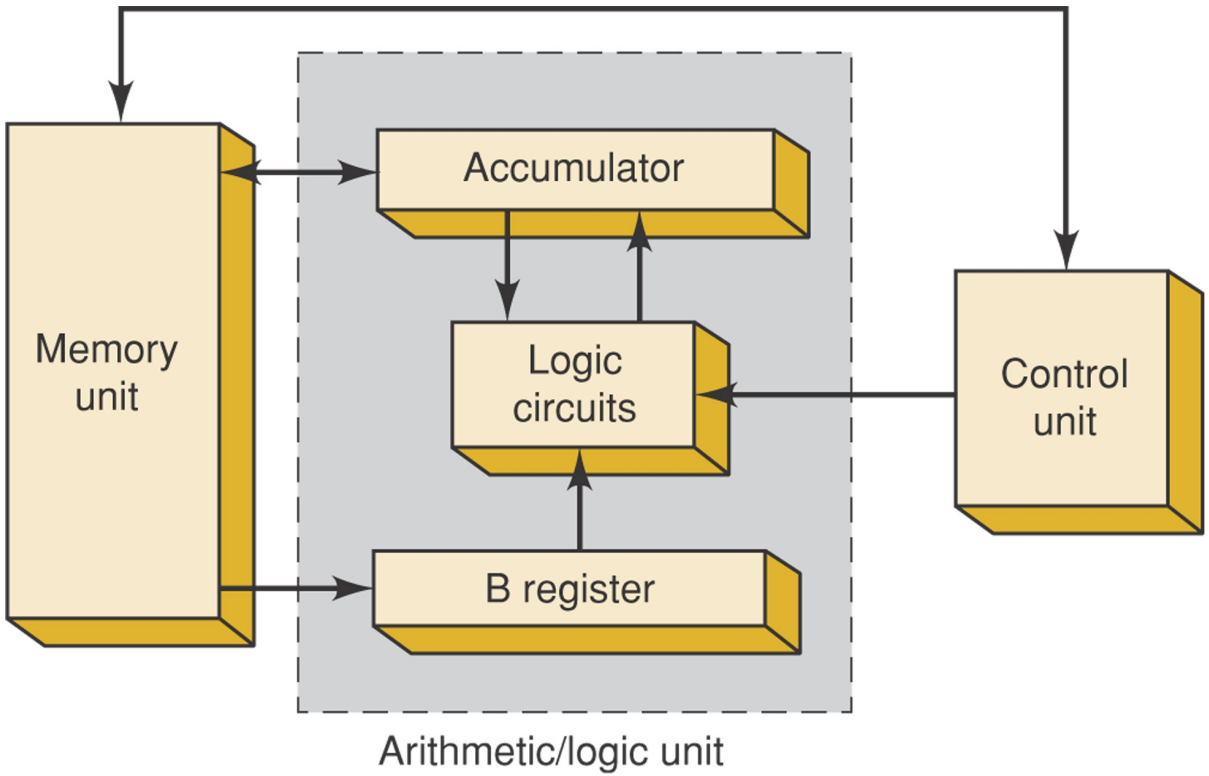
Representation of Signed Numbers in the 2's-complement System



A Four-bit Number Circle



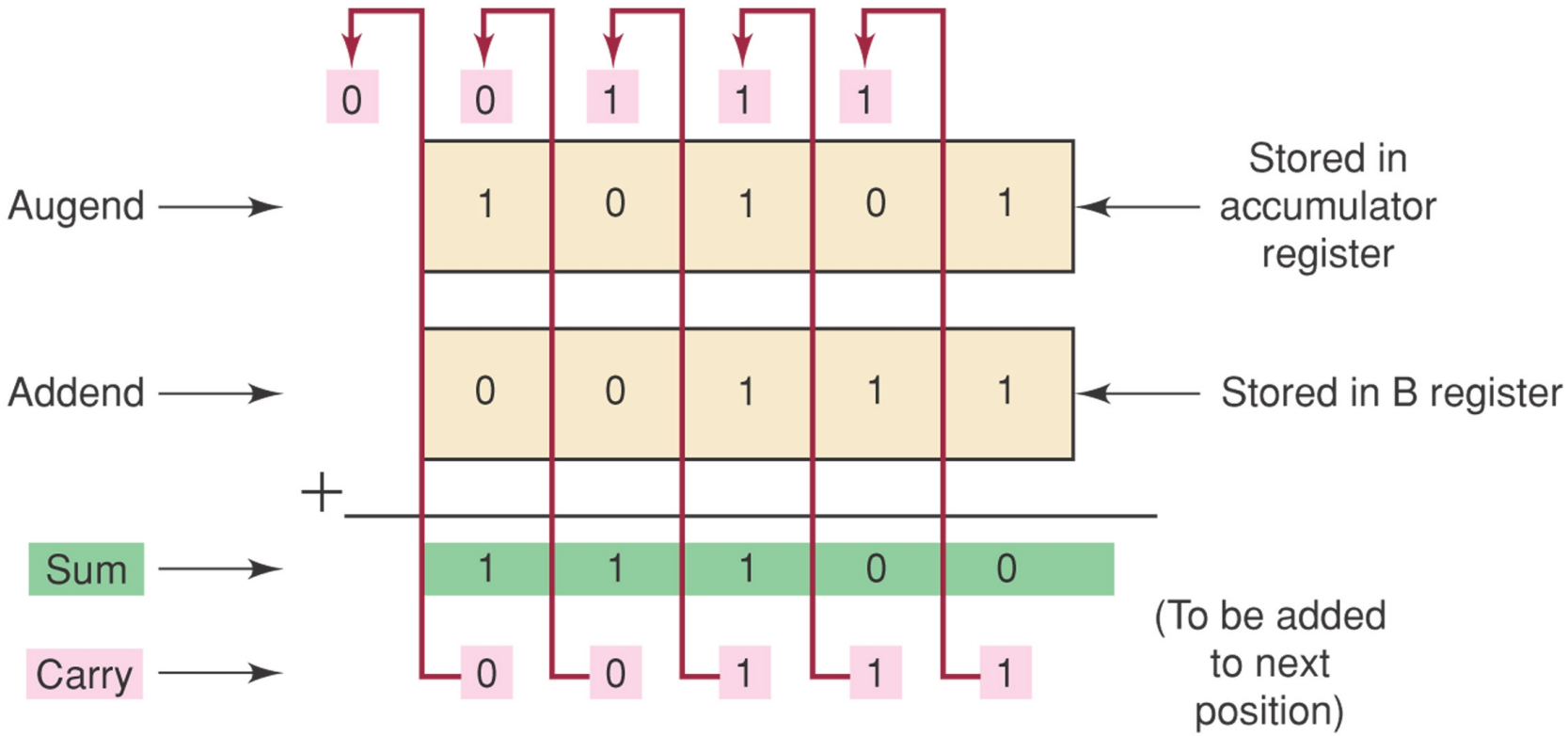
Functional Parts of an ALU



Addition in the 2's Complement System

- Perform normal binary addition of magnitudes
- The sign bits are added with the magnitude bits
- If addition results in a carry of the sign bit, the carry bit is ignored
- If the result is positive then it is in pure binary form
- If the result is negative then it is in 2's complement form
- If the result of the addition exceeds the number of magnitude bits an overflow occurs

Typical Binary Addition Process



Addition in the 2's Complement System: Case 1 and 2

Case 1: Two positive numbers

+9 →	0	1001	(augend)
+4 →	0	0100	(addend)
	0	1101	(sum = +13)

↑ sign bits

Case 2: Positive number and smaller negative number

sign bits ↓

+9 →	0	1001	(augend)
−4 →	1	1100	(addend)
1	0	0101	

↑ This carry is disregarded; the result is 00101 (sum = +5).

Addition in the 2's Complement System: Case 3 and 4

Case 3: Positive number and larger negative number

$$
\begin{array}{rl}
-9 \rightarrow & 10111 \\
+4 \rightarrow & 00100 \\
\hline
 & 11011 \quad (\text{sum} = -5)
\end{array}
$$

↑ negative sign bit

Case 4: Two negative numbers

$$
\begin{array}{rl}
-9 \rightarrow & 10111 \\
-4 \rightarrow & 11100 \\
\hline
\not{1} & 10011
\end{array}
$$

↑ sign bit

↑ This carry is disregarded; the result is 10011 (sum = −13).

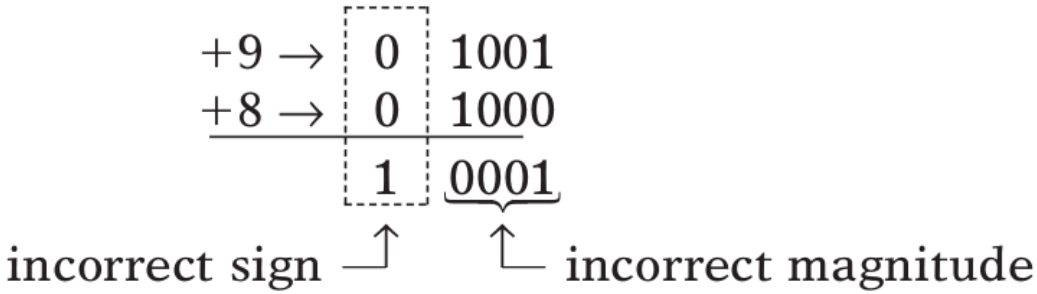
Addition in the 2's Complement System: Case 5

Case 5: Equal and opposite numbers

$$
\begin{array}{rl}
-9 \rightarrow & 10111 \\
+9 \rightarrow & 01001 \\
\hline
0 \quad 1 & 00000
\end{array}
$$

Disregard; the result is 00000 (sum = +0).

Overflow during Addition



- Overflow can occur only when two positive or two negative numbers are being added, and it always produces an incorrect result
- Overflow can be detected by checking to see that the sign bit of the result is the same as the sign bits of the numbers being added

Subtraction in the 2's Complement System

- The number subtracted (subtrahend) is negated
- The result is added to the minuend
- The result of the addition represents the difference
- If the result of the addition exceeds the number of magnitude bits an overflow occurs

BCD Addition (1)

- If the sum of each decimal digit is less than 9 then the operation is the same as normal binary addition
- If the sum of each decimal digit is greater than 9 then a binary 6 is added, this will always cause a carry
- 4 bits are used per digit
- BCD subtraction – a more complicated operation – will not be discussed here

BCD Addition (2)

- Seems to work just as normal binary addition

```
  5      ____      (BCD for 5)
 +4   + ____      (BCD for 4)
 --     ====
  9      ____      (BCD for 9)
```

- Another example

```
  45     ____ ____   (BCD for 45)
 +33   + ____ ____   (BCD for 33)
 --     ==== ====
  78     ____ ____   (BCD for 78)
```

- Both examples do not create carries

BCD Addition (3)

- Seems to work just as normal binary addition

```
  5   0101    (BCD for 5)
 +4  +0100    (BCD for 4)
 ---  ----
  9   1001    (BCD for 9)
```

- Another example

```
  45     0100  0101    (BCD for 45)
 +33    +0011  0011    (BCD for 33)
 ---    ----  ----
  78     0111  1000    (BCD for 78)
```

- Both examples do not create carries

BCD Addition (4)

- But does not work if sum > 9

```
  6   0110    (BCD for 6)
 +7  +0111    (BCD for 7)
 --  ----
 13   1101    invalid code group

      0110     (BCD for 6)
     +0111     (BCD for 7)
     ----
      1101     invalid sum
      0110     add 6 for correction
 0001 0011     (BCD for 13)
```

- 0110 is added to the invalid sum, produces carry

Hexadecimal Arithmetic (1)

Hex addition:

- Add the hex digits in decimal
- If the sum is 15 or less then express it directly in hex digits
- If the sum is greater than 15 then subtract 16 and carry 1 to the next position

```
  58
+4B
----
```

Hexadecimal Arithmetic (2)

- Hex subtraction – use the same method as for binary numbers (hex $\rightarrow$ binary $\rightarrow$ 2's complement $\rightarrow$ hex)
- A quicker method to get 2's complement is illustrated below
- Find 2's complement for 73A:

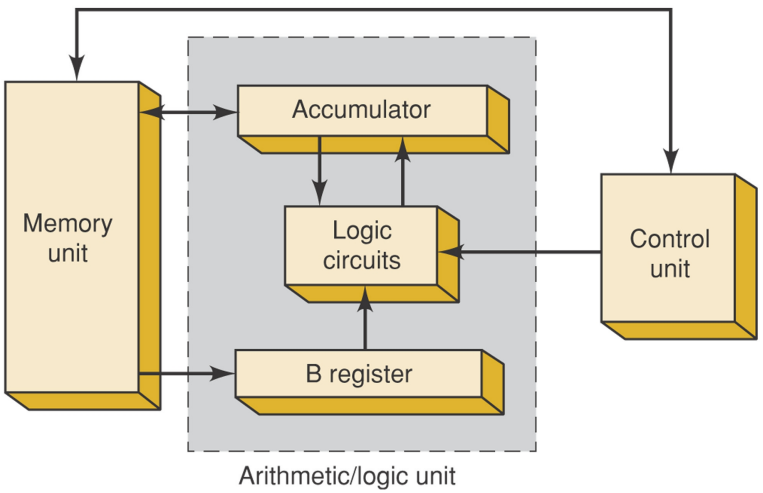
```
  F     F     F
-7    -3    -A
  8     C     5
              +1
-------------------
  8     C     6
```

Hexadecimal Representation of Negative Numbers

- Negative number: Highest bit is 1 (binary)
- Positive number: Highest bit is 0 (binary)
- If the MSD (Most Significant Digit) in a hex number is 8 or greater then the number is negative
- If the MSD is 7 or less then the number is positive

Arithmetic Circuits (1)

- An arithmetic/logic unit (ALU) accepts data stored in memory and executes arithmetic and logic operations as instructed by the control unit
- Contains at least two flip-flop registers:
  - B register
  - Accumulator register



Arithmetic Circuits (2)

- Typical sequence of operations:
  - Control unit is instructed to add a specific number from a memory location to a number stored in the accumulator register
  - The number is transferred from memory to the B register
  - Number in the B register and accumulator register are added in the logic circuit, with sum sent to the accumulator for storage
  - The new number remains in the accumulator for further operations or can be transferred to memory for storage
- Register accumulates the sums when performing successive additions

Parallel Binary Adder

- The A and B variables represent two 5-bit numbers to be added
- The C variables are the carries
- The S variables are the sum bits