ECE2700J SU24 Mid RC

Wenyue Li
6/19/2024

Combinational Circuit

- Combinational Circuit
  - A digital circuit whose output depends only upon the ***present combination*** of its inputs
  - Output changes only when any of the inputs changes

- As oppose to Sequential Circuit which is
  - A digital circuit whose output depends not only upon the present input values, but also the history of input and output values
  - Have feedbacks from outputs
  - More complicated and difficult to analyze

Combinational Circuit

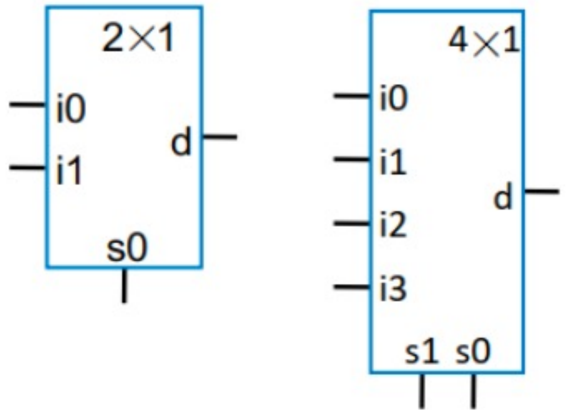
How to Design a Combinational Circuit

Step		
Step 1	**Capture** the function	By **truth table/ equation**
Step 2	**Convert** to equations	**Simplify** by K-map or Boolean algebra
Step 3	**Implement** as a logic circuit	With **logic gates and blocks** mentioned in slides Pay attention to the **format** of the blocks

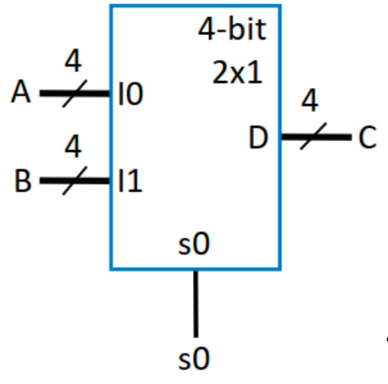
JOINT INSTITUTE
交大密西根学院

Combinational Circuit

MUX



2x1 Mux / 4x1 Mux



4bit 2x1 Mux (with **bus**)

Important Elements (Don't forget anyone)

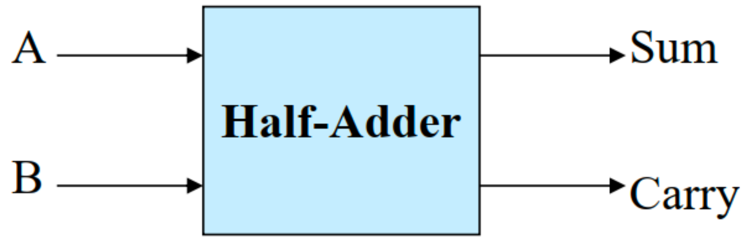
- Block Label: m-bit N*1 mux
- Input label: i0,i1,i2,…
- Select Signal label: s0, s1,s2
- Output label: d

If you are required to draw one single Mux, remember:

- Input/Output signals: a, b, c, F

Combinational Circuit

Half adder (1 bit)



Important Elements

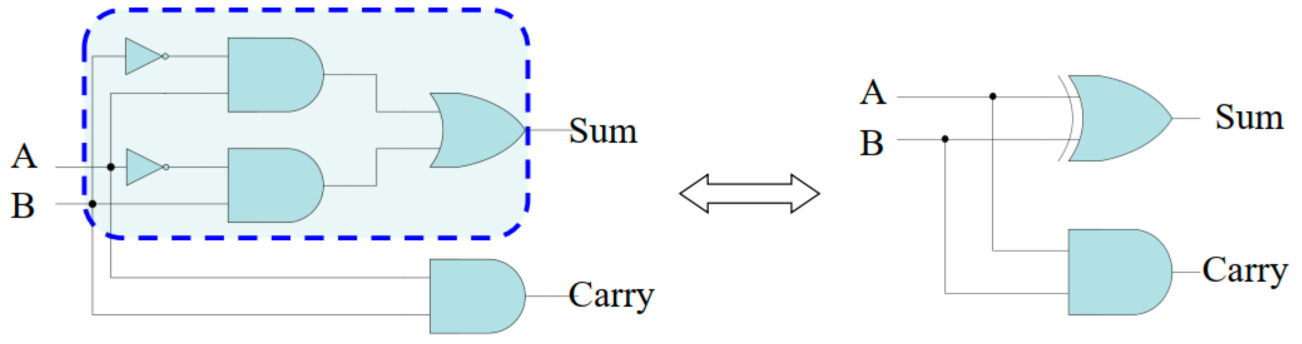
- Block Label: HA
- Input label: A, B
- Output label: Sum(s), Carry(c)

- **Equations:**

$$Sum = A'B + AB' = A \oplus B$$

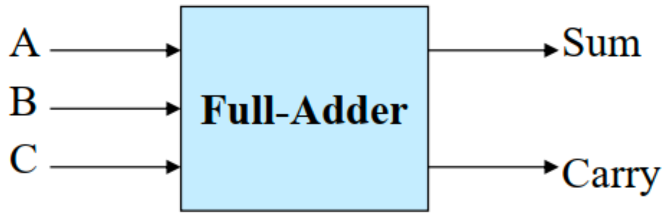
$$Carry = AB$$

Inner Structure



Combinational Circuit

Full adder (1 bit)



- **Equations:**

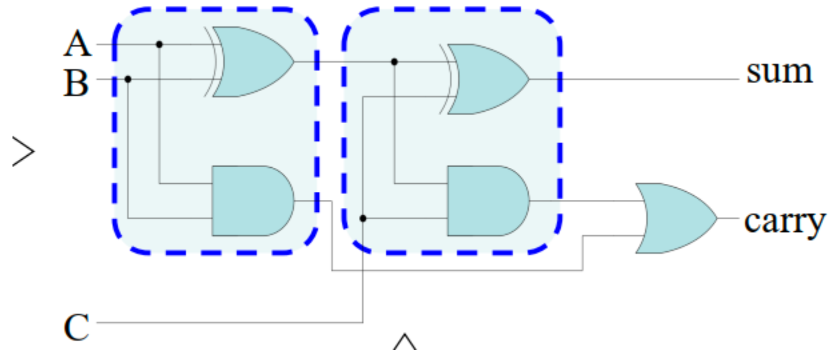
$$Sum = (A \oplus B) \oplus C$$

$$Carry = (A \oplus B)C + AB = AB + AC + BC$$

Important Elements

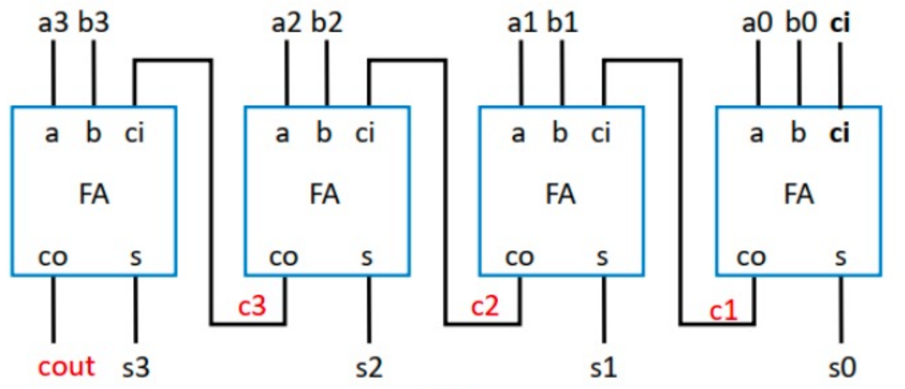
- Block Label: FA
- Input label: A, B, C
- Output label: Sum(s), Carry(c)

Inner Structure:

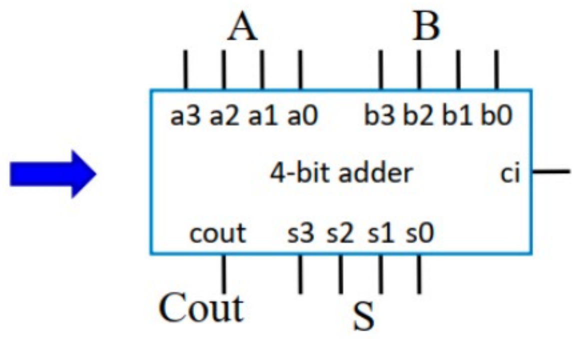


Combinational Circuit

Carry Ripple adder (N bits)



(a)



(b)

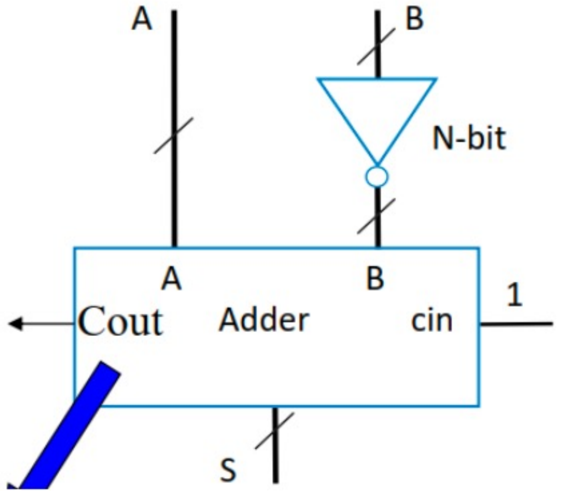
Important Elements

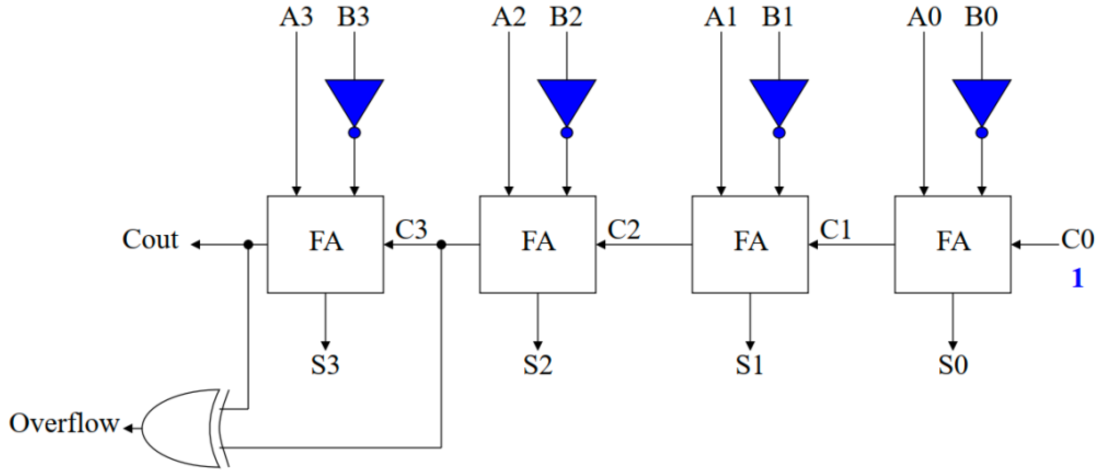
- Block Label: N-bit adder
- Input label: A(a0, a1,…), B(b0,b1,…), ci(1 bit)
  (You can use **bus** here)
- Output label: S (s0, s1,…), Cout (1 bit)

Combinational Circuit

Two's complement Subtractor (N-bits)

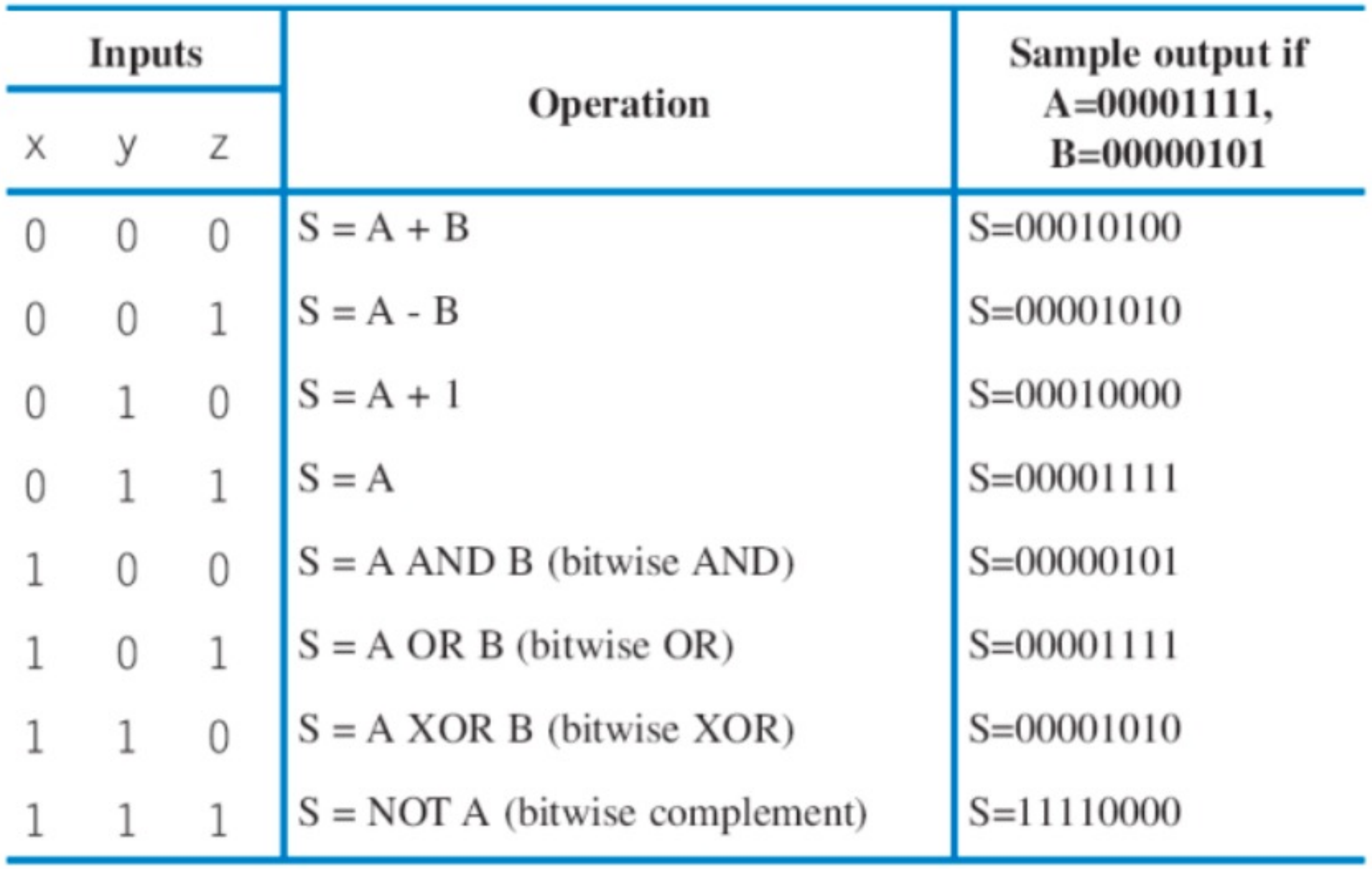
- $A - B = A - (two's\ complement of B) = A + invert\_bits(B) + 1$





Combinational Circuit

ALU



Inputs			Operation	Sample output if A=00001111, B=00000101
x	y	z		
0	0	0	S = A + B	S=00010100
0	0	1	S = A - B	S=00001010
0	1	0	S = A + 1	S=00010000
0	1	1	S = A	S=00001111
1	0	0	S = A AND B (bitwise AND)	S=00000101
1	0	1	S = A OR B (bitwise OR)	S=00001111
1	1	0	S = A XOR B (bitwise XOR)	S=00001010
1	1	1	S = NOT A (bitwise complement)	S=11110000

JOINT INSTITUTE
交大密西根学院

Combinational Circuit

Encoder & Decoder

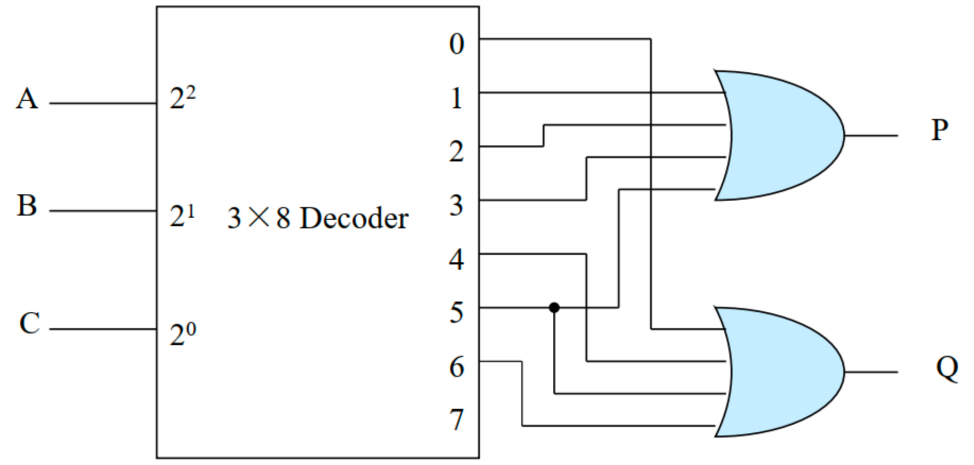
Encoder

- Each input is given a binary code: Among all inputs, only 1 input equals to 1

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Decoder

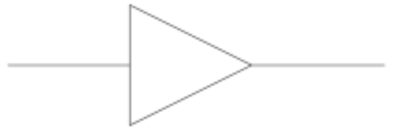
- Convert binary code to one high output: Among all outputs only 1 output equals to 1
- Minterm Generator (easy to draw the form of sum of minterms)



Combinational Circuit

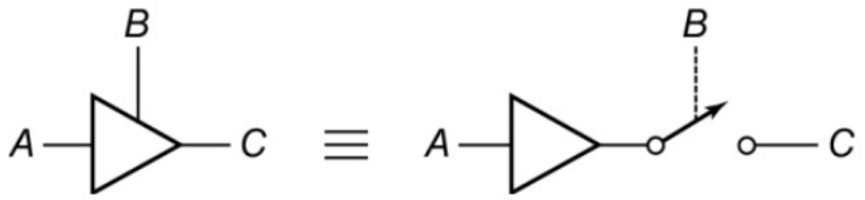
Buffer

- A buffer is a one directional transmission logic device
  - somewhat like a NOT gate without complementing the binary value
  - amplify the driving capability of a signal
  - insert delay
  - Protect input from output

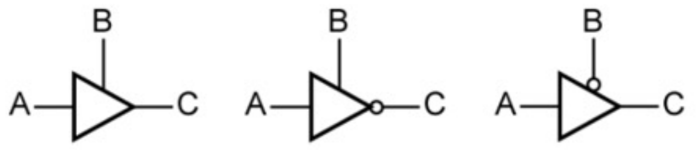


Combinational Circuit

Tri-state Buffer



- **Input: A; Output: C; Control signal: B**
  - B = 0, A = x -> C = Z (**high impedence, no voltage**)
  - B = 1, A = 0 -> C = 0 (off)
  - B = 1, A = 1 -> C = 1 (on)
- Other types of tri-state buffer



B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

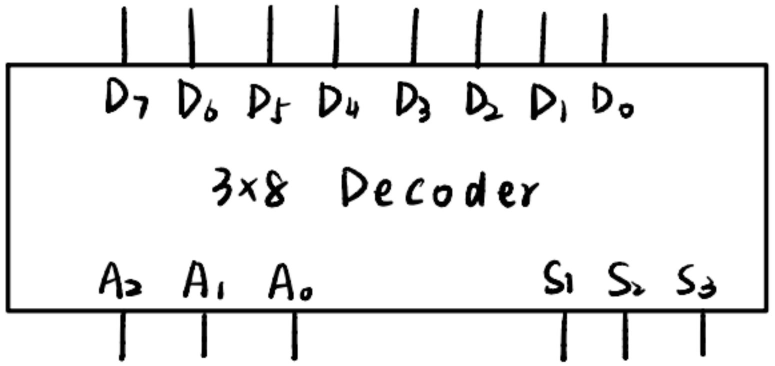
B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0

B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z

Combinational Circuit

Exercise

[EX1] Please use 3×8 decoders with control signal s_1, s_2, s_3 (shown below) to construct a 4×16 decoder. Notice only when $s_1 = 1$, $s_2 = 0$, $s_3 = 0$, it acts like a decoder.



Combinational Circuit

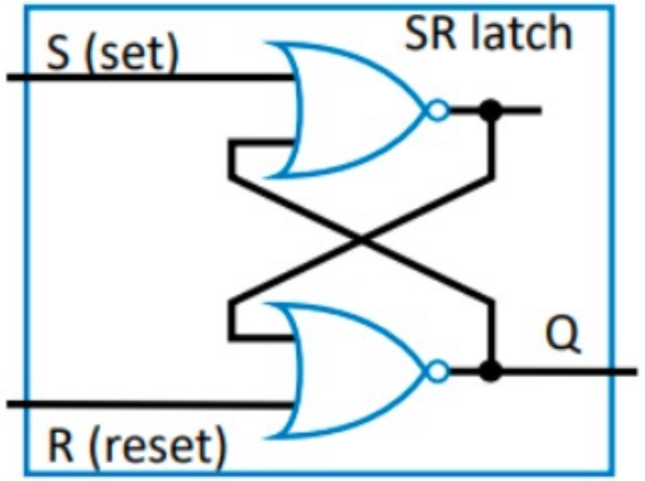
Exercise

[EX2] Design a three-switch, one-light system, requiring changing any switch will control light to change from 0 -> 1 or 1 -> 0. Realize this with MUX.

(Initial state: A = 0, B = 0, C = 0, Light is off.)

Sequential Circuit

SR Latch

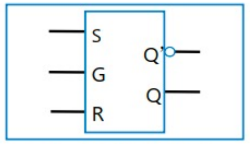


$S(t)$	$R(t)$	$Q(t)$	$Q(t+\Delta) \longrightarrow Q^+$	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	not allowed
1	1	1	X	

- Mind the not allowed case
- RS Latch: reversed version of SR Latch
- Gated SR Latch: add enable signal G to SR Latch

S	R	Q	Q^+	
0	0	0	X	not allowed
0	0	1	X	
0	1	0	1	set
0	1	1	1	
1	0	0	0	reset
1	0	1	0	
1	1	0	0	hold
1	1	1	1	

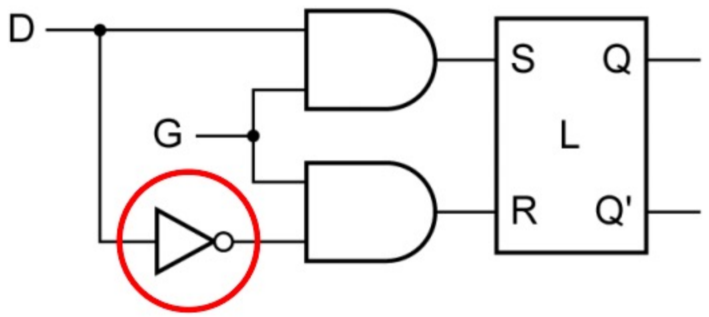
G S R	Q^+
0 x x	Q; Latch locked



Gated SR latch symbol

Sequential Circuit

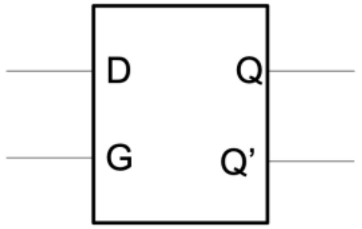
Gated D Latch



Characteristic Table

G	D	Q^+
1	0	0
1	1	1
0	X	Q

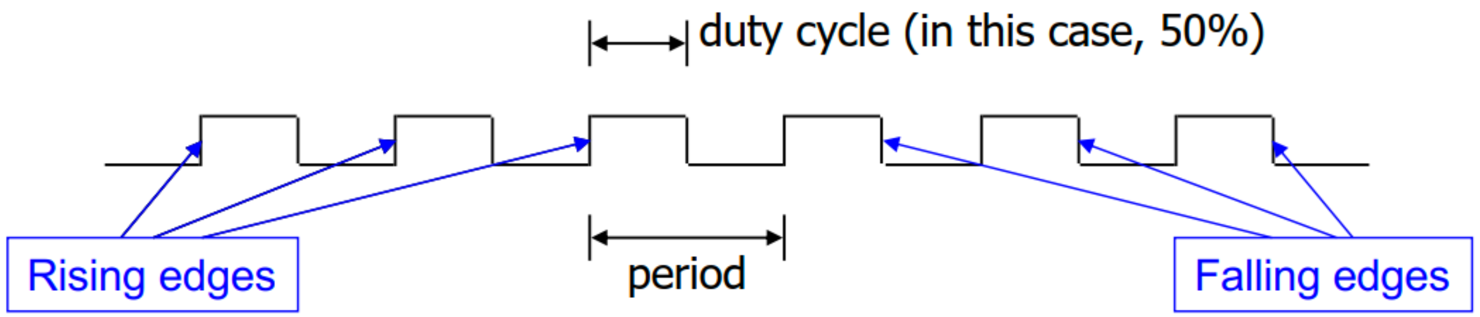
- Eliminate not allowed case
- Temporary storage
- Transparent latch



D latch symbol

Sequential Circuit

Clock

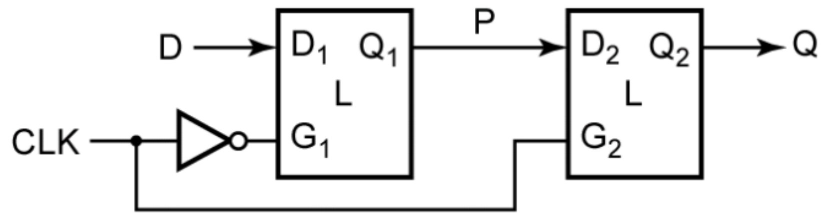


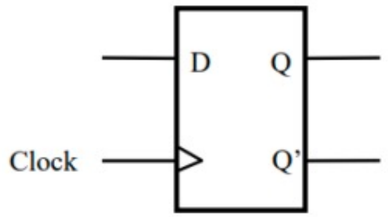
- ***Clock period***: time interval between pulses
- ***Clock cycle***: one such time interval
- ***Clock frequency***: 1/period

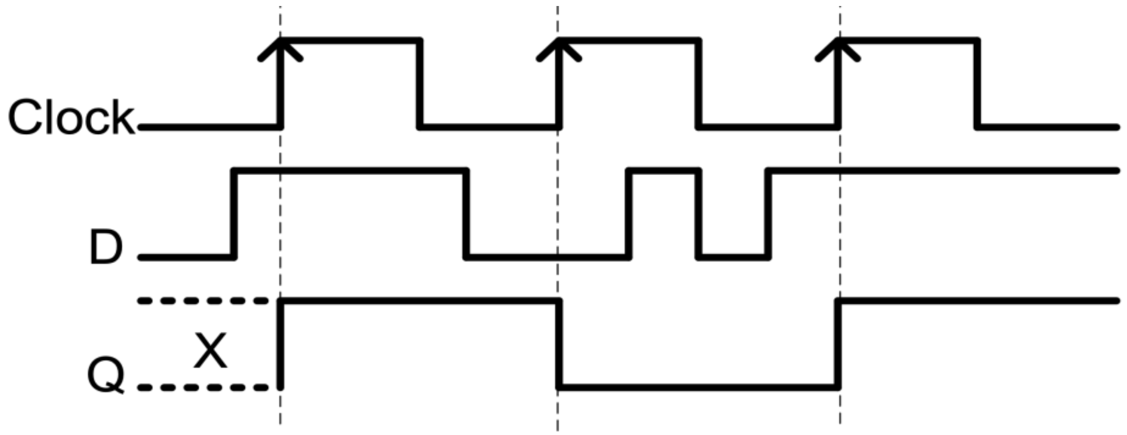
Sequential Circuit

Rising-Edge Triggered D Flip Flop

- Rising-edge triggered Master-Slave D flip flop



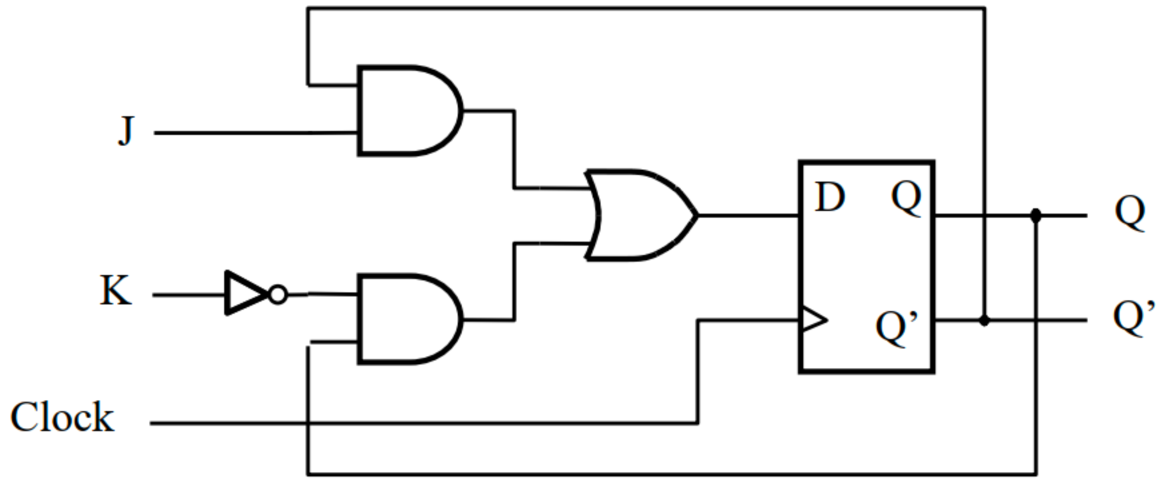


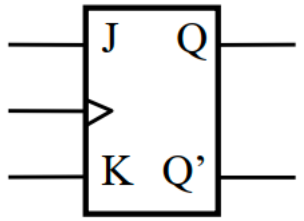


clock	D	Q^+
↑	0	0
↑	1	1
0	X	Q
1	X	Q

Sequential Circuit

Rising-Edge Triggered J-K Flip Flop



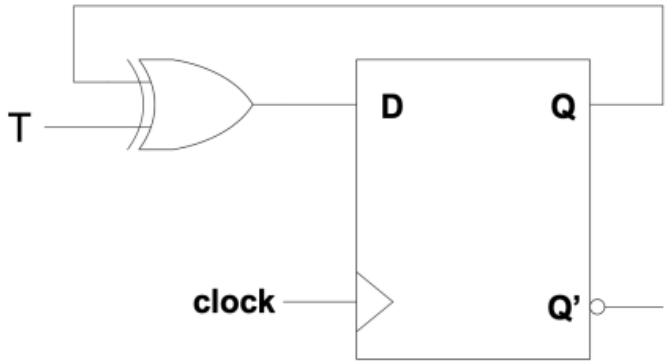


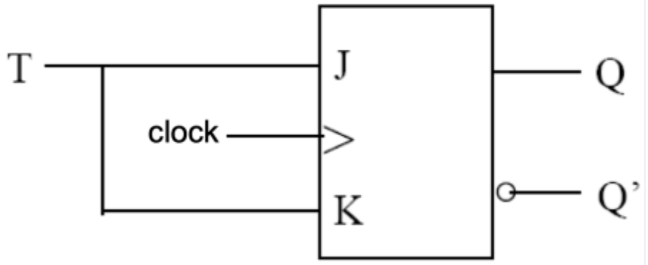
J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q'

Characteristic equation:
$Q^+ = JQ' + K'Q$

Sequential Circuit

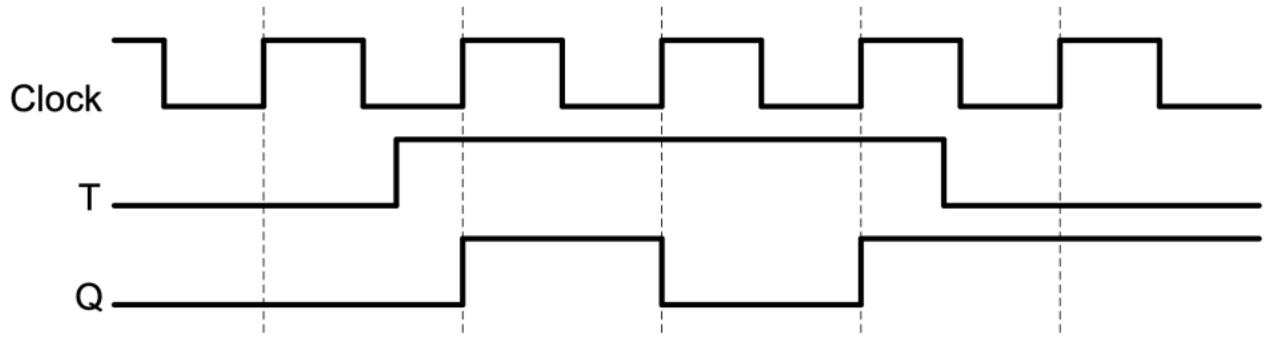
Rising-Edge Triggered T Flip Flop

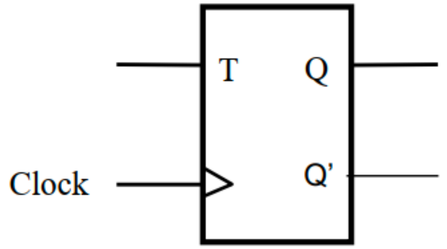




clock	T	Q^+
↑	0	Q
↑	1	Q'

Characteristic equation:
$Q^+ = T'Q + TQ' = T \oplus Q$





Sequential Circuit

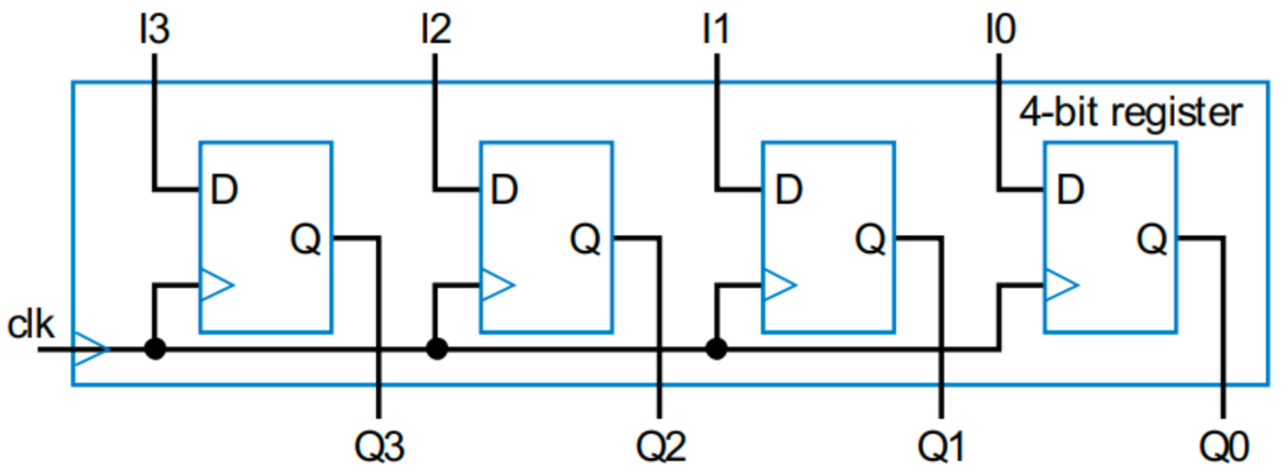
Latch & Flip Flop

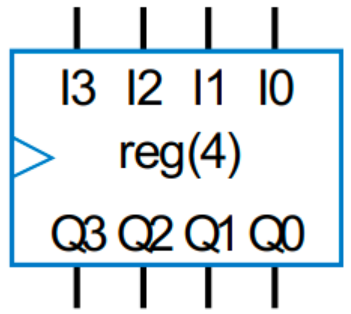
- **Both are** storage elements **in sequential circuits**
- **Flip flop**
  - **edge-sensitive**, the input matters only at active edges (rising or falling)
  - behaviors are **synchronous** to the clock signal
- **Latch**
  - **level-sensitive**, the input matters whenever control has active level (high or low)
  - behaviors are **asynchronous** to the clock signal

Sequential Circuit

Register

- ***Register***: multiple flip-flops sharing clock signal

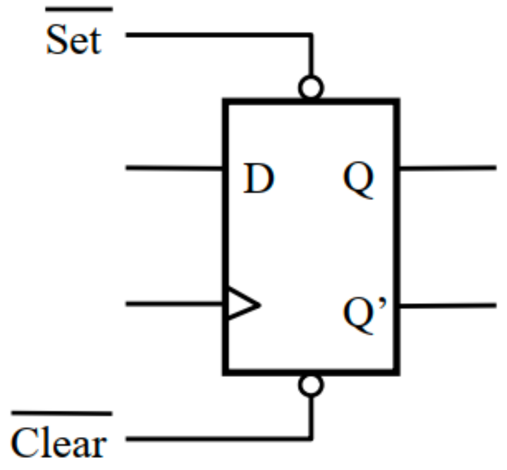


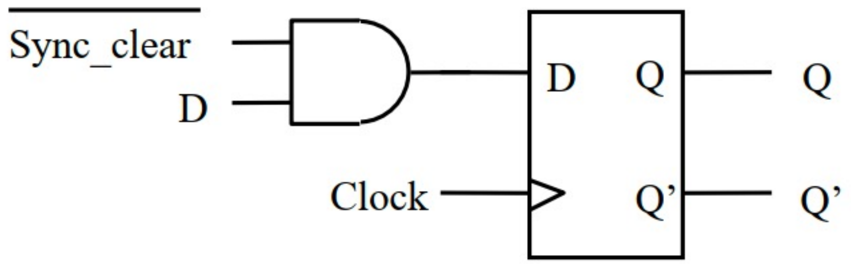


Sequential Circuit

Control Inputs for Flip Flops

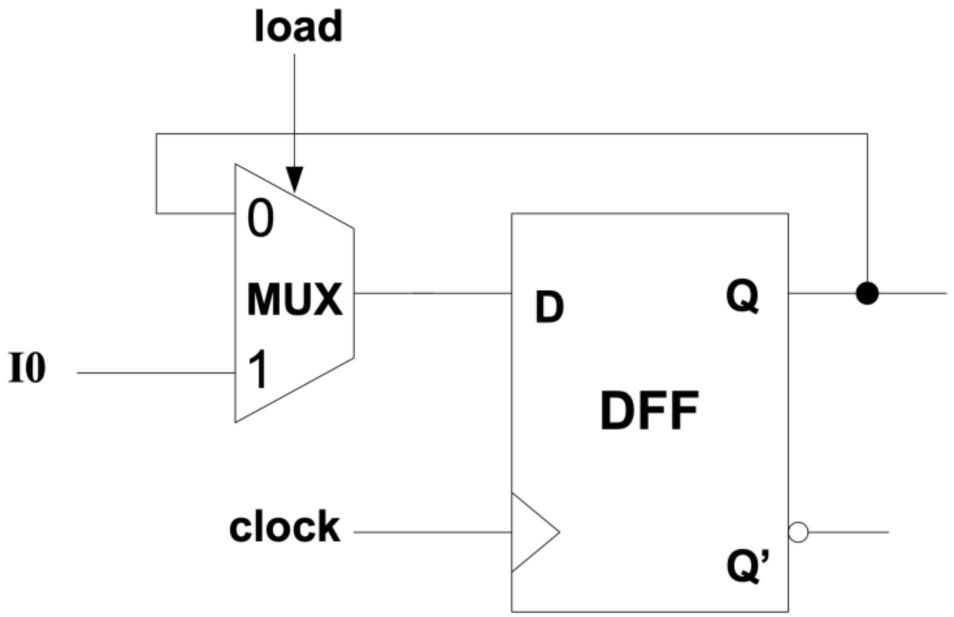
- Asynchronous:
  - control signals do not depend on the clock signal
- Synchronous:
  - control signals depend on the clock signal
- Active low:
  - It controls when it's low
- Active high:
  - It controls when it's high

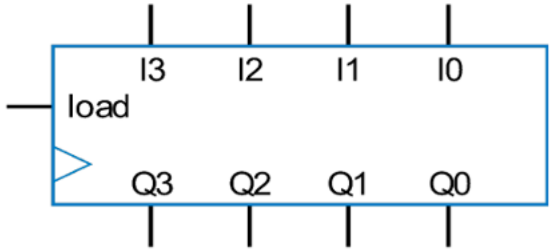




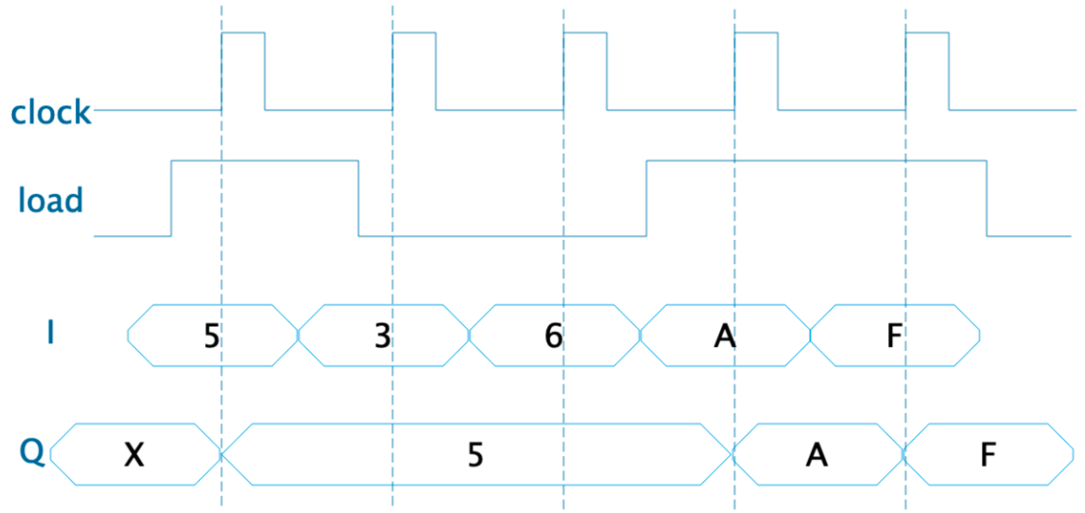
Sequential Circuit

Register with Synchronous Parallel Load





- D flip flop with active low synchronous Clear



Counter

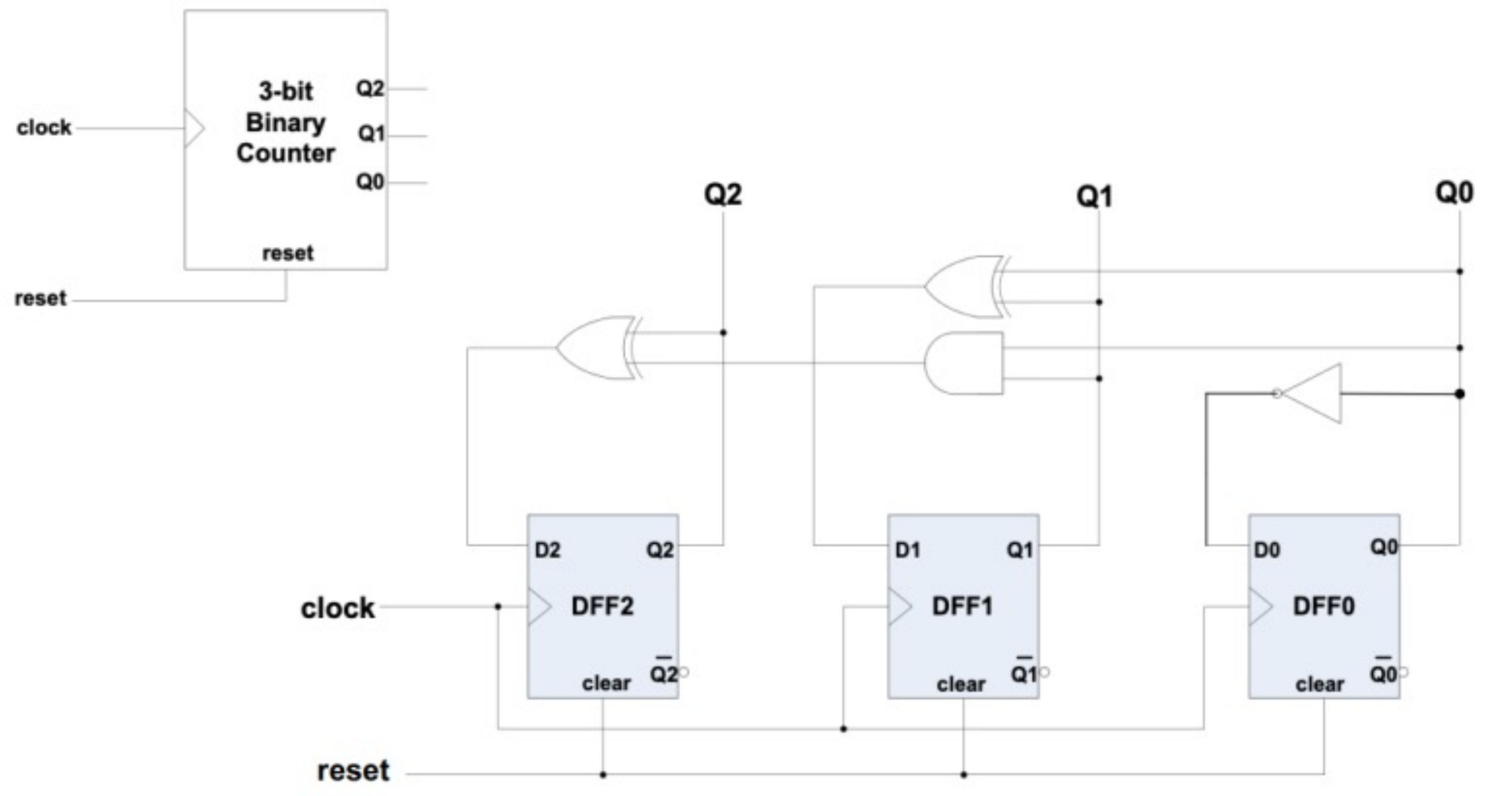
Definition of counters

- A digital circuit that counts in different format: binary, decimal, signed, unsigned…
- An n-bit binary counter can count up to 2^n-1 numbers
- An n-bit binary counter consists of n flip-flops
- Count up or count down, increment or decrement once per clock cycle – counting number of clock cycles

JOINT INSTITUTE
交大密西根学院

Counter

Synchronous 3-bit Binary Counter with D Flip-Flop



Combinational & Sequential part

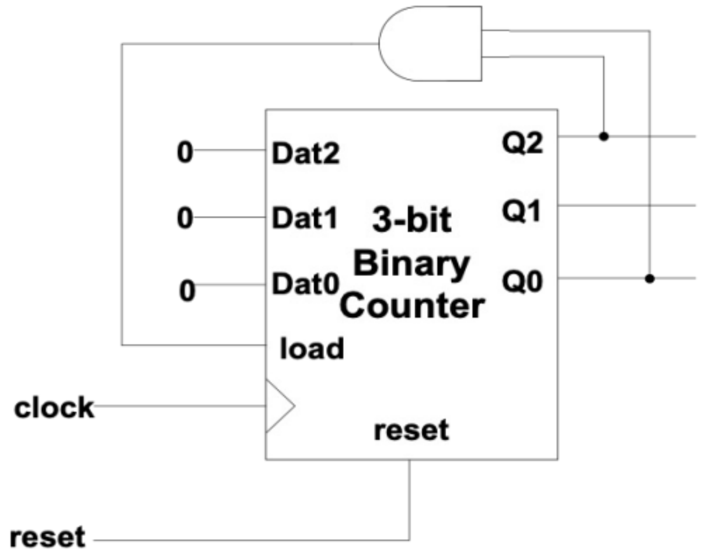
The combinational part is designed **based on truth table**

Present State			D flip flop input		
Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Counter

Load in counter

- **Load (Parallel Load):**
  - Numbers can be loaded into the counter anytime when the load input is high, thus the count sequence can be customized



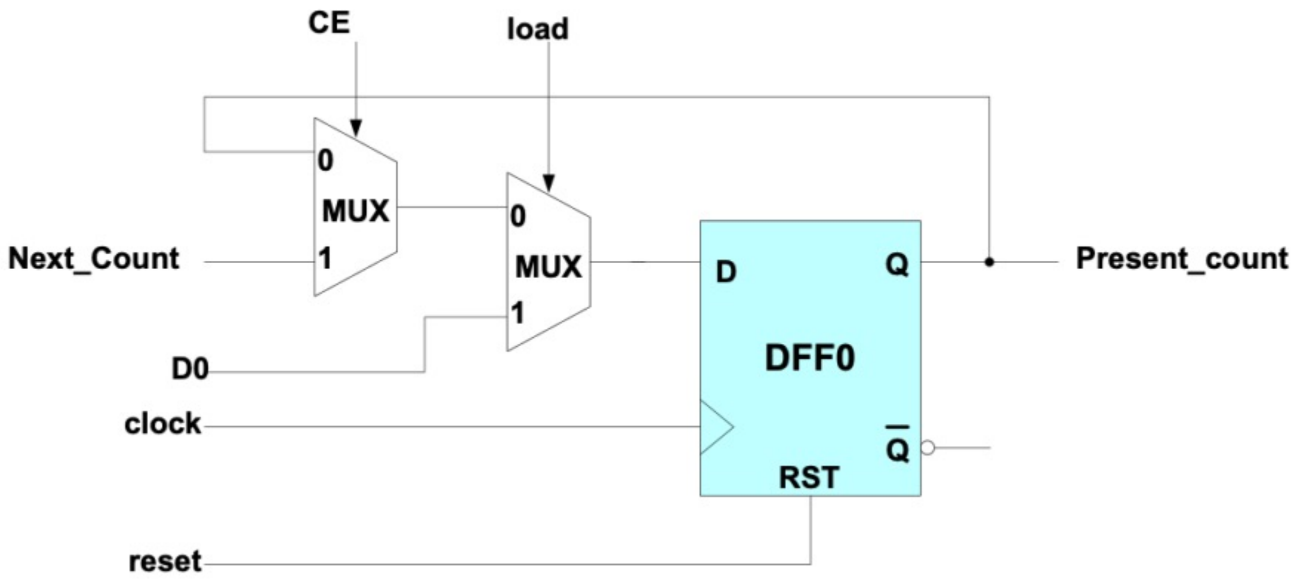
Q2	Q1	Q0	$Q2^+$	$Q1^+$	$Q0^+$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0		X	
1	1	1			

- load = Q2Q0, initial number = 000

Counter

Counter with Count Enable and Load

- **Count Enable (CE) :**
  - when CE = 1, counter counts
  - when CE = 0, counter holds the values
- Implementation of 1-bit of a counter with both CE and load:

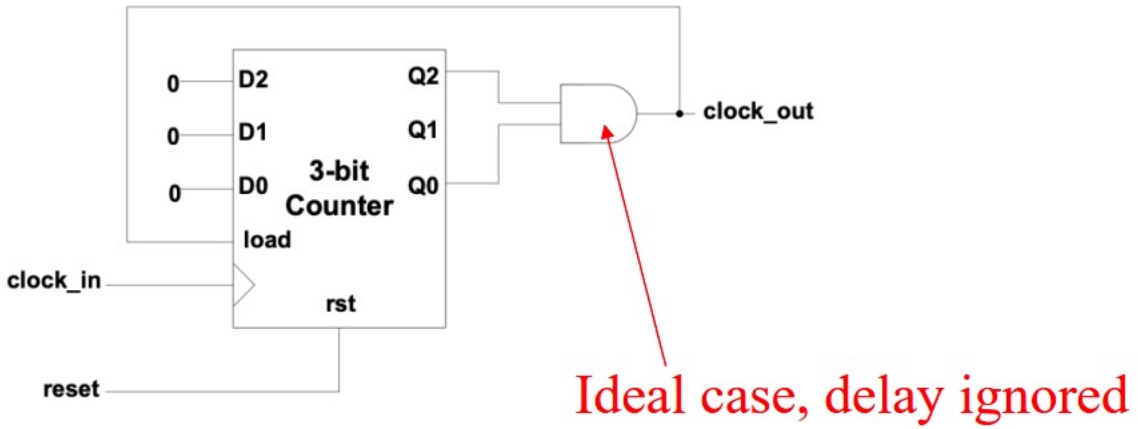


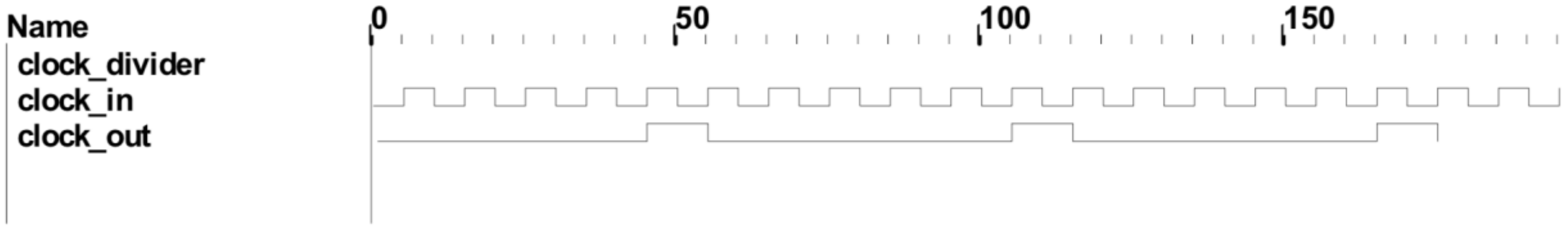
Reset > Load > Count Enable

Counter

Clock Divider: Example divide by 6

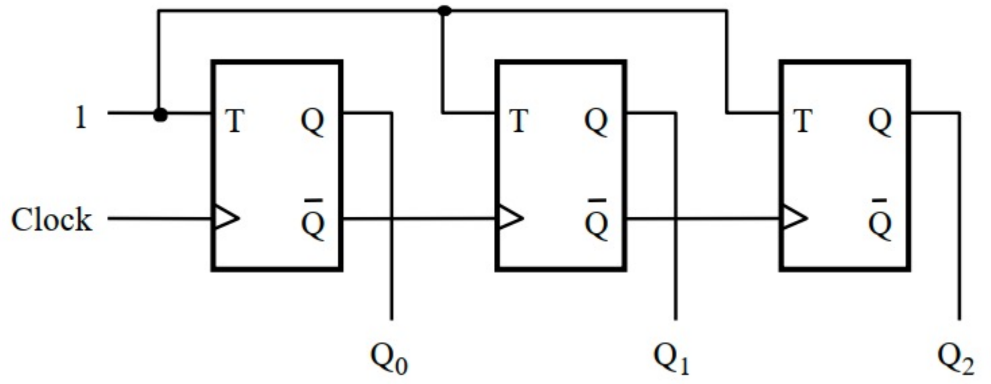
- **Frequency** is divided - – Generates clock signal with bigger clock cycle
  - Countering sequence: 000→001→010→011→100→101→000
  - The output should be a logic “1” whenever Q2 and Q0 are high
    - clock_out = load = Q2 & Q0

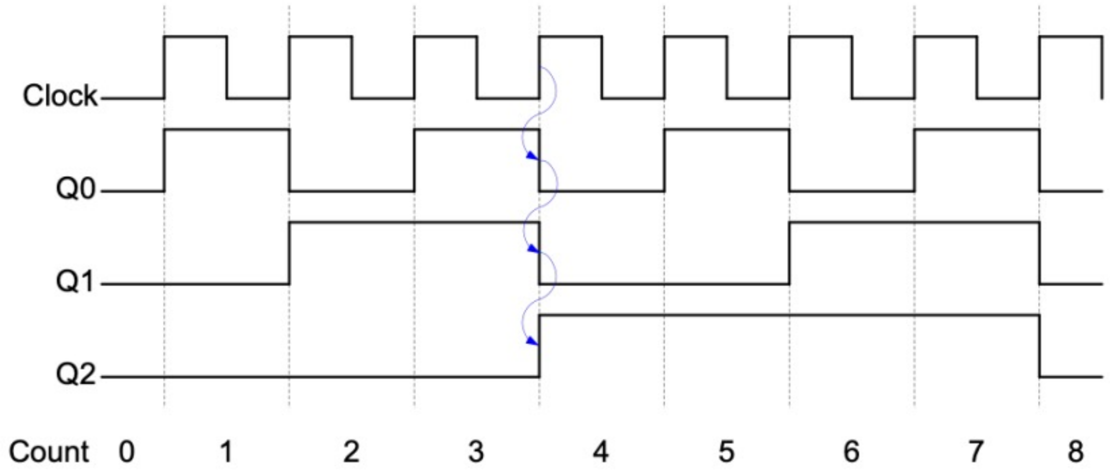




Counter

Asynchronous 3-bit Binary Counter with T Flip-Flop

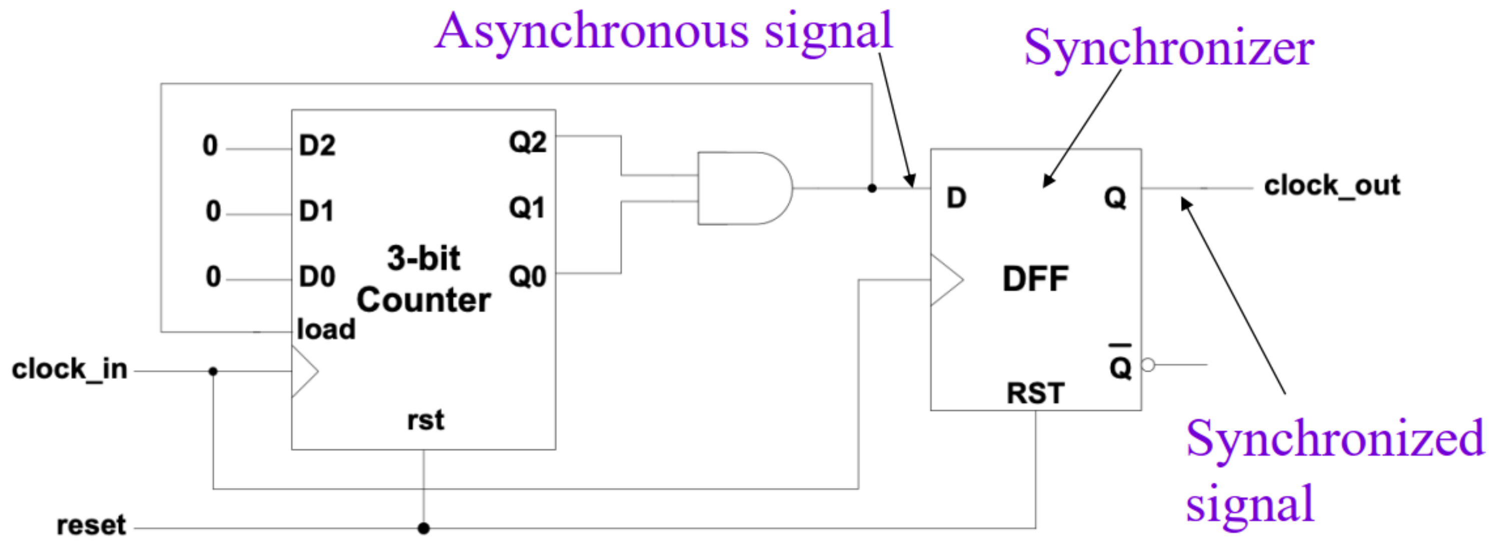




- Get familiar with the samples in slides
- Probably you will be asked some models in the choice questions
- Problem: asynchronous counter：
  – Delays caused by each stage – timing issues
- **Clock skew**: difference in arrival time of clock signal between two sequentially adjacent registers

Counter

Synchronizer



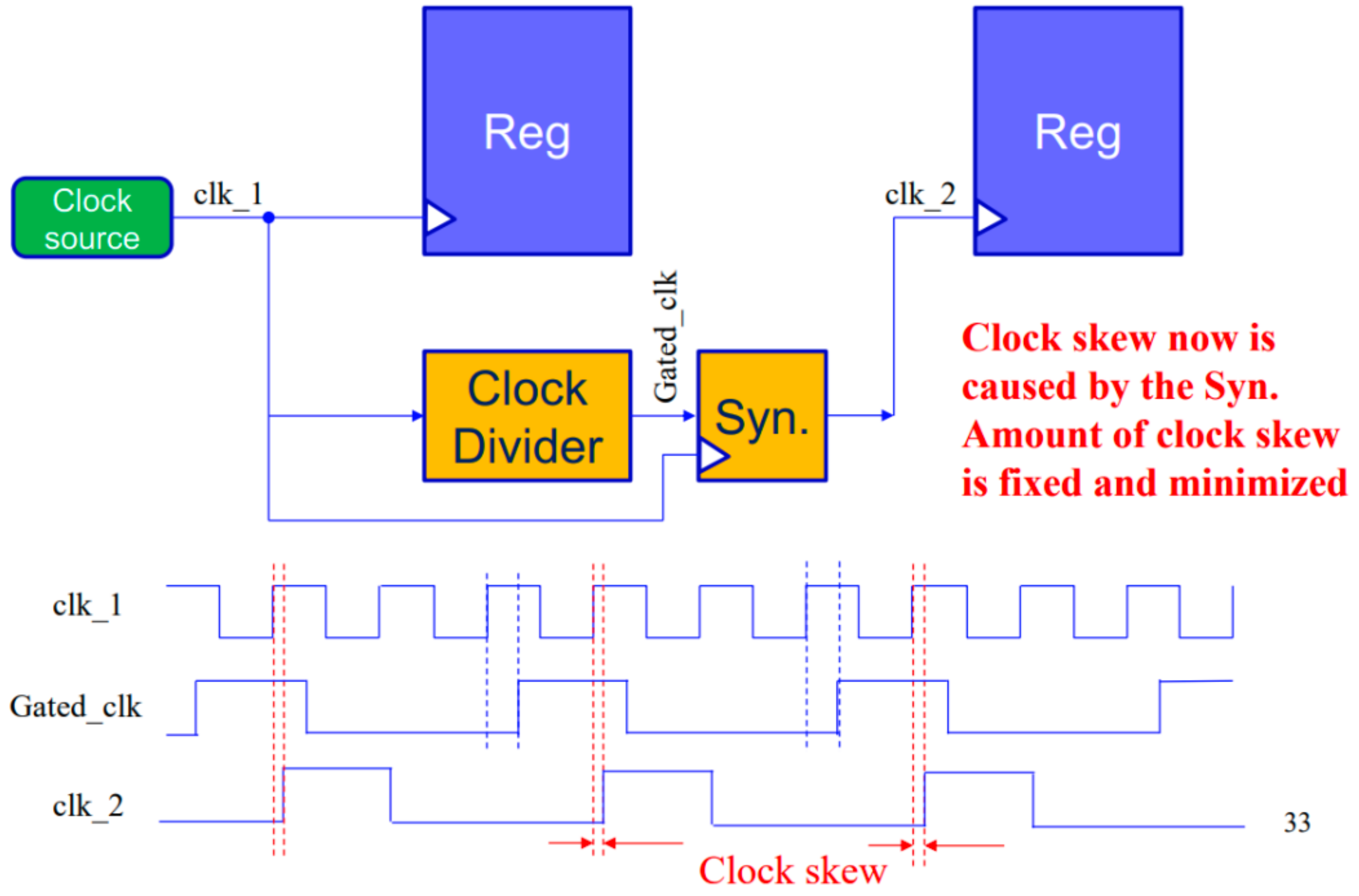
- Synchronize the output by a D flip-flop
- Synchronizer delays the input asynchronous signal by up to 1 clock cycle

– But **aligns** the synchronized signal with clock

– Clock skew **is not completely removed**, only reduced

Counter

Synchronized Clock



Proper frequency is given to the synchronizer

Counter

More Counters

Ring counter

- A ring counter is a circular shifter with only one FF being set at any time

$$\begin{tikzcd}
\text{start} \arrow[r] & 1000 \arrow[r] & 0100 \arrow[d] \\
& 0001 \arrow[u] & 0010 \arrow[l]
\end{tikzcd}$$

Current				Next			
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	1
0	0	0	1	1	0	0	0
all other inputs				X			

D3 = Q0
D2 = Q3
D1 = Q2
D0 = Q1

Johnson counter

- Counting sequence

  1000 → 1100 → 1110 → 1111 → 0111 → 0011 → 0001 → 0000 → …
- In each state transition, only one bit has to be changed
  - less state transition effort
  - but less counting state too

BCD counter

Q3	Q2	Q1	Q0	$Q3^+$	$Q2^+$	$Q1^+$	$Q0^+$
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X			
. . .							
1	1	1	1				

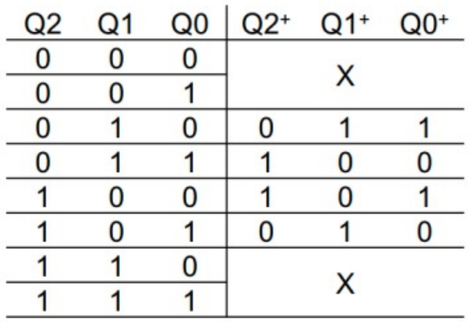
Counter

Design Counters

General step:

- Build truth table for each Q+ based on output Q
- Modify the combinational part according to the truth table

Exercise: Design a counter to Count from 2-5 (Hint use load)



Q2	Q1	Q0	$Q2^+$	$Q1^+$	$Q0^+$
0	0	0		X	
0	0	1			
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0		X	
1	1	1			

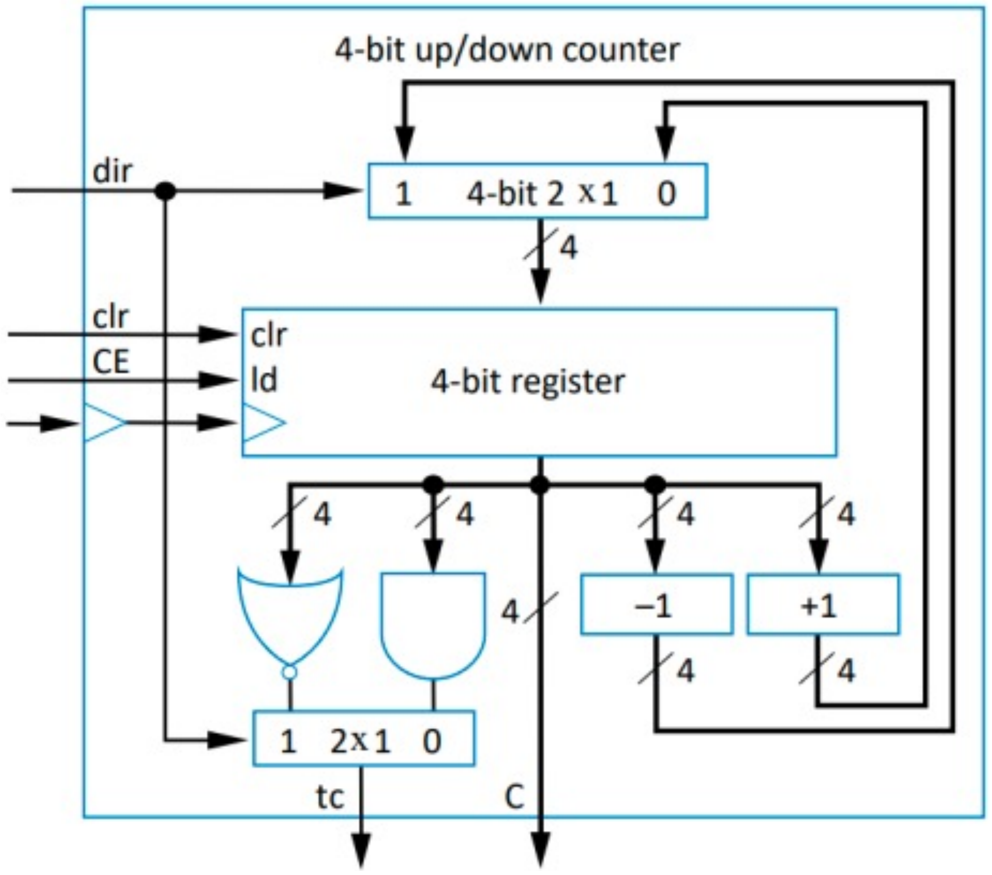
Counter

Design Counters

- Design a 4-bit counter that outputs all the prime numbers within 0-15 in ascending order. Draw the schematic.

Counter

Up/Down-Counters



Incrementer/Decrementer can be used directly

Learn to figure out the function of each input from the circuit