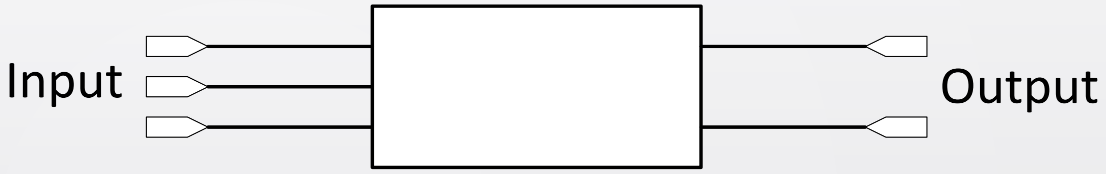
Tutorial 3 Flip-Flops

COMP2120B Computer organization

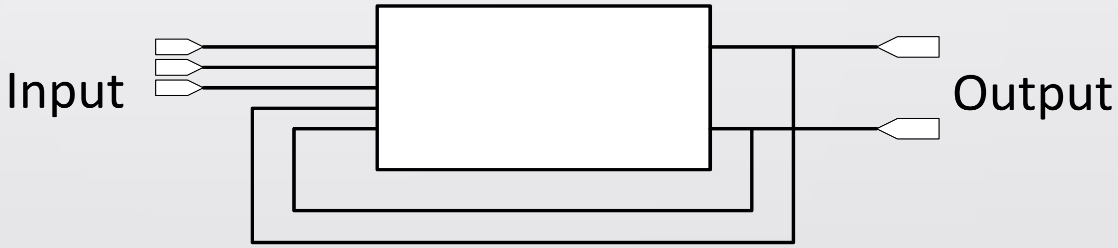
Kevin Lam (yklam2)

Overview

- A combinational circuit generates outputs base on the current input only.



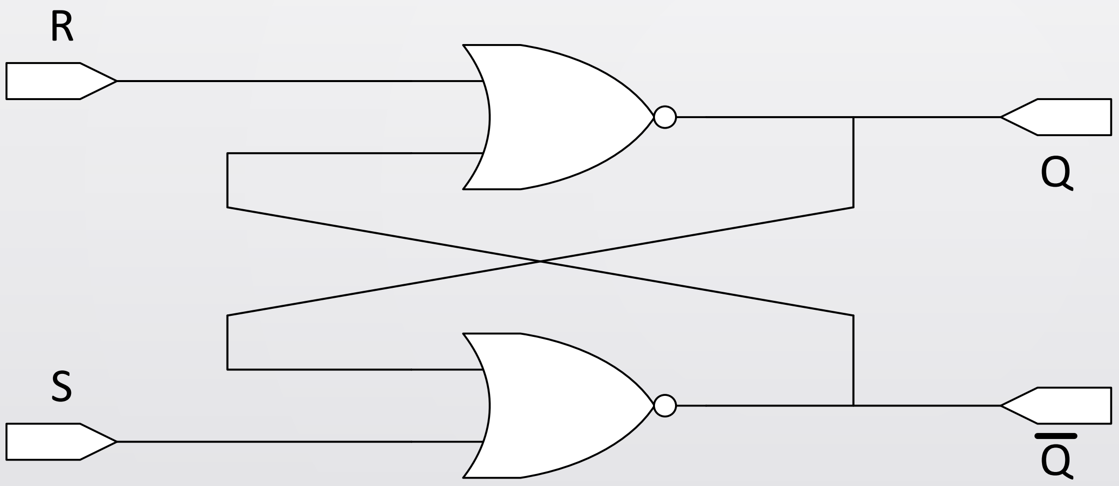
- A sequential circuit generates outputs base on the current and the past input.



  - Past input is usually captured using the previous output.
- Flip-flop is one of the simplest form of sequential circuit, we start with the S-R latch.

Analyzing S-R latch

- The S-R latch and the corresponding truth table.



Observation
As output Q and $\overline{Q}$ will be served as input immediately,
The output will be updated again for the highlighted cases.

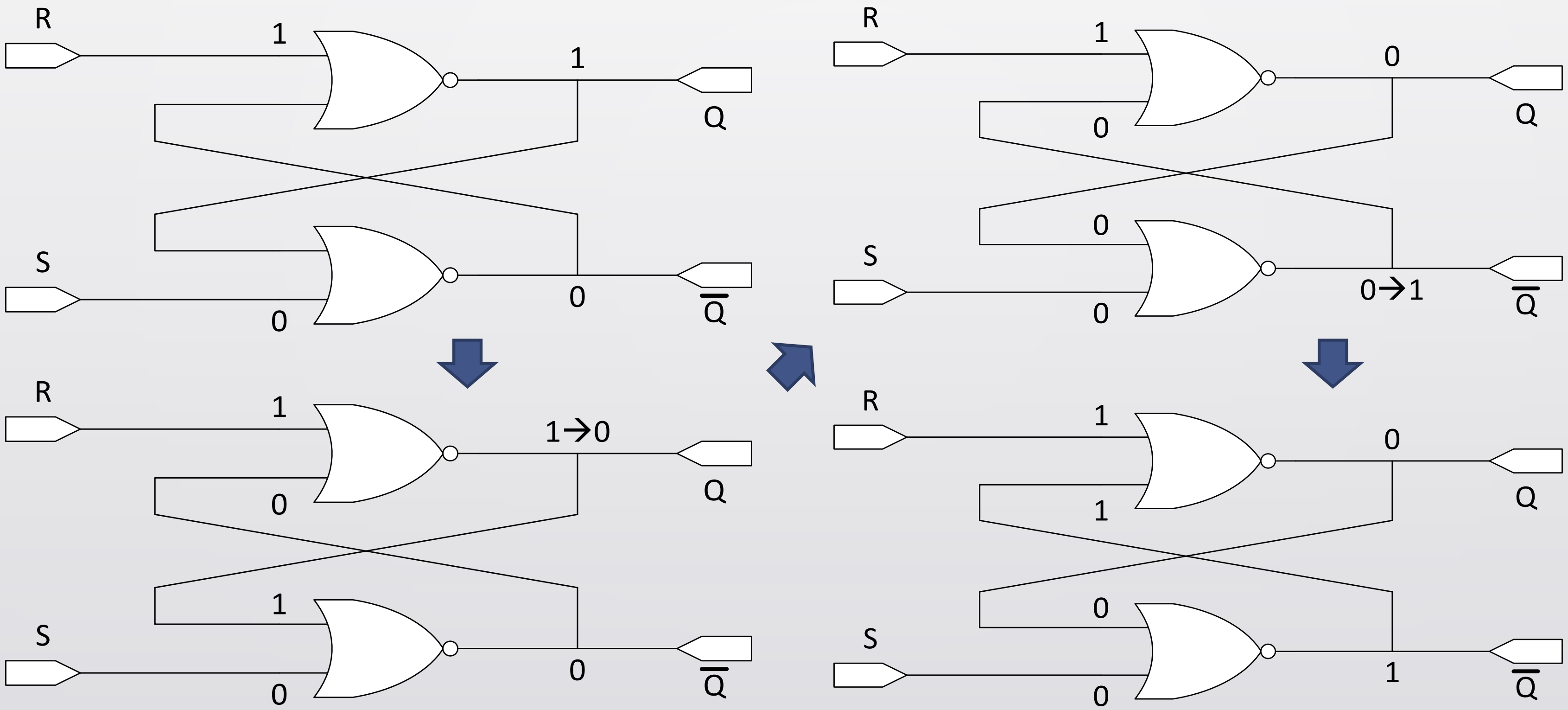
Input				Output	
S	R	Q	$\overline{Q}$	Q	$\overline{Q}$
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Stable state

Input				Output(1)		Output(2)		Output(3)	
S	R	Q	$\overline{Q}$	Q	$\overline{Q}$	Q	$\overline{Q}$	Q	$\overline{Q}$
0	1	1	0	0	0	0	1	0	1

State won't change anymore

- Let's consider one of the case

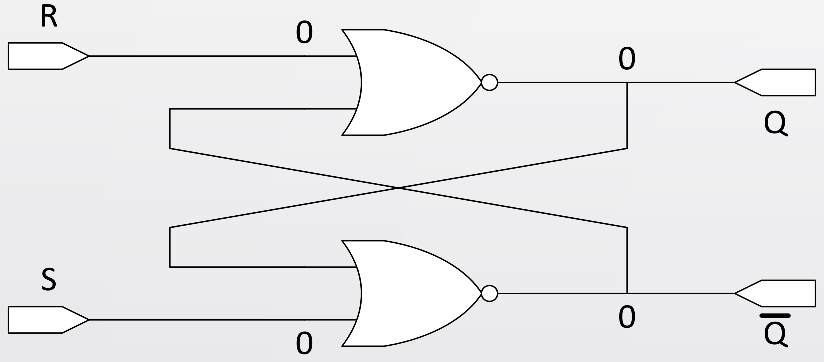


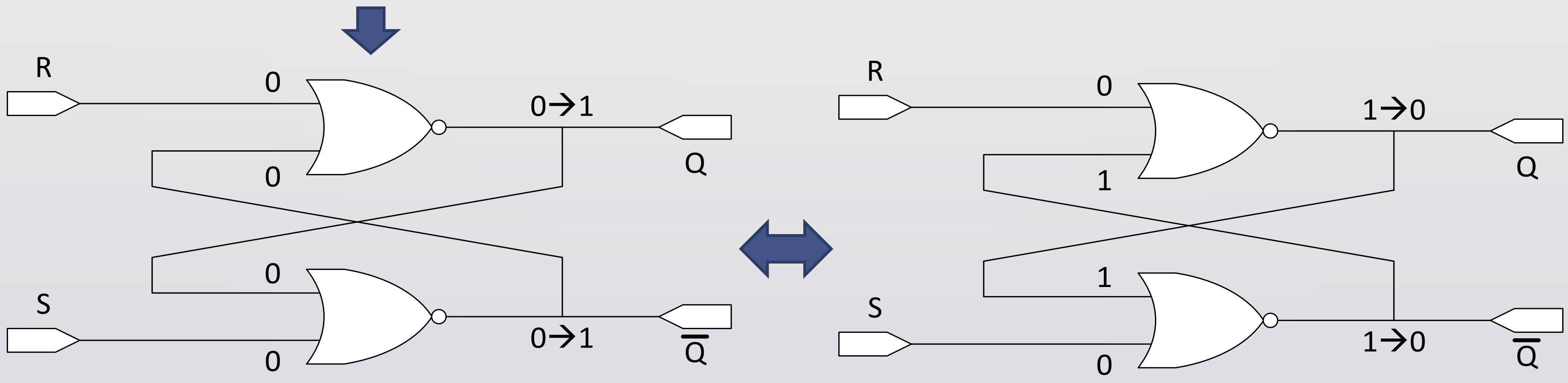
Unstable state

Input				Output(1)		Output(2)		Output(3)	
S	R	Q	$\overline{Q}$	Q	$\overline{Q}$	Q	$\overline{Q}$	Q	$\overline{Q}$
0	0	0	0	1	1	0	0	1	1

State will never be stable

- Let's consider another case





S-R Latch – stable states

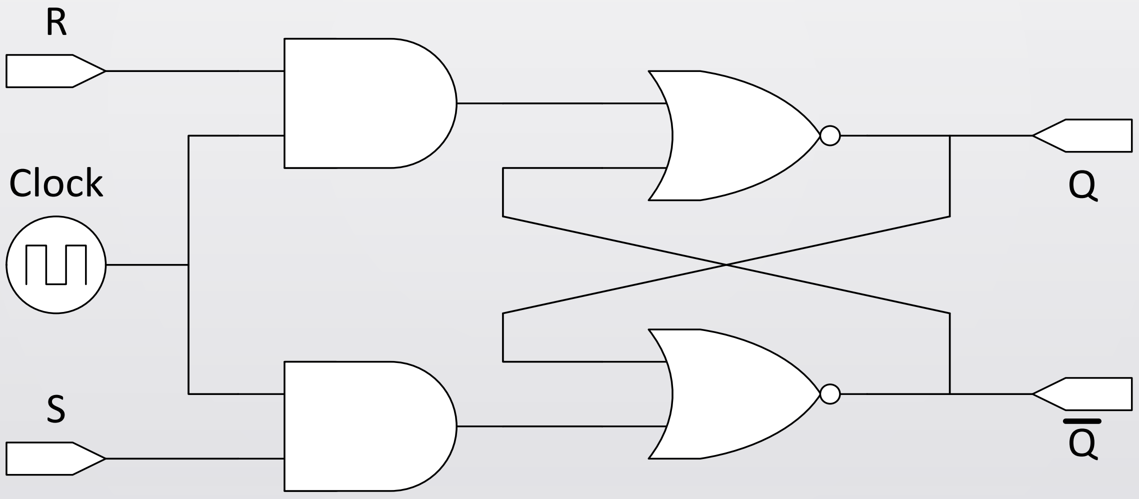
- We can repeat the analysis to find all stable states.
    - The latch is unstable only when S and R were set to zero AND when $Q = \overline{Q}$.
        - However, this happens only when we set both S and R to 1.
    - If we avoid setting both S and R to 1, we can always avoid reaching an unstable state.
        - Q and $\overline{Q}$ are then always complementary.
- The result can then be further summarized.

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	-

Input				Output	
S	R	Q	$\overline{Q}$	Q	$\overline{Q}$
0	0	0	0	Unstable	
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	Unstable	
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

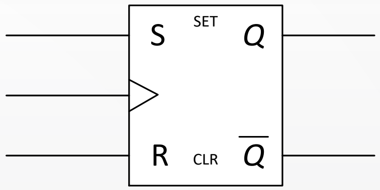
Clocked S-R Latch

- To synchronize S-R state with clock pulse, the clock signal is used to control when the input of S and R should be taken.



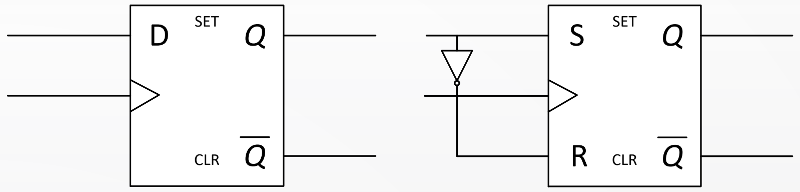
Only when clock is 1, the 1 in S or R could be sent to the latch

S-R Flip-flops

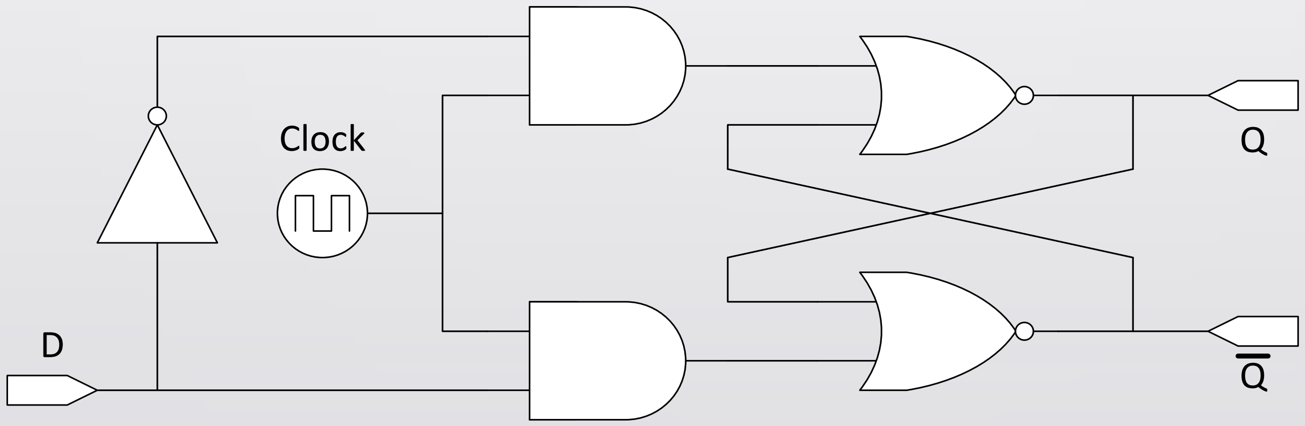


- **Bistable** device, could be used to store 1 bit of data.
- Built from **clocked latches**.
- Two outputs, which are always the complements of each others. (Q and $\overline{Q}$)
- Two other typical flip-flops are D flip-flop and J-K flip-flop.

D flip-flop

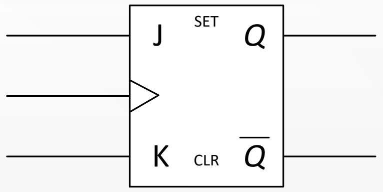


- Single input (D).
- Also called data flip-flop or delay flip-flop.
- Using a NOT gate, forcing only one of the input for the S-R latch to be 1.

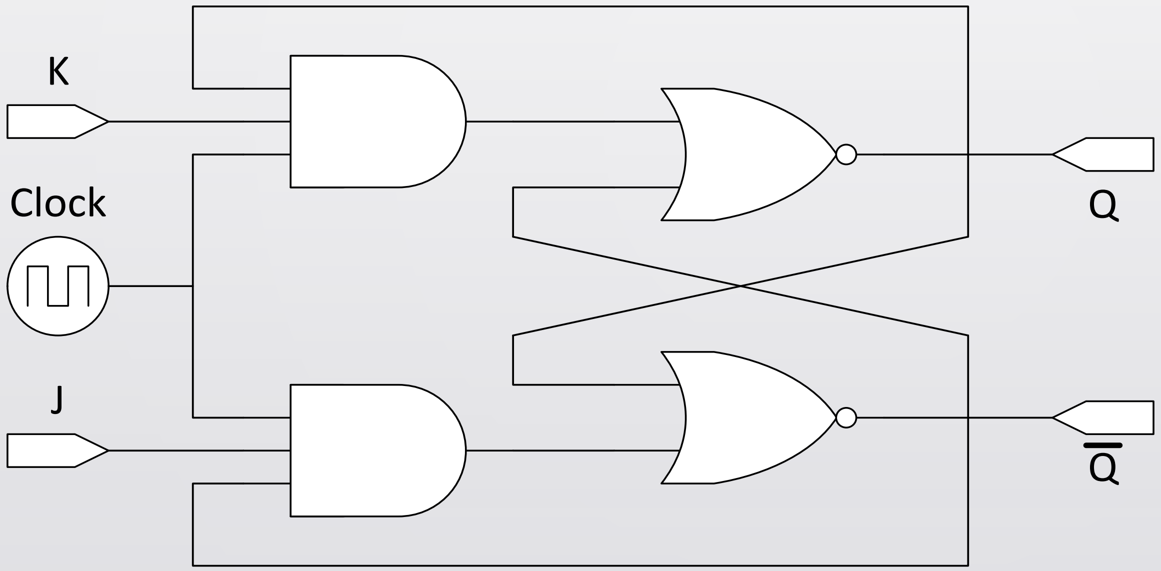


D	Q_{n+1}
0	0
1	1

J-K flip flop



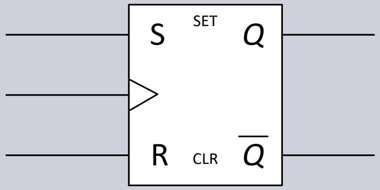
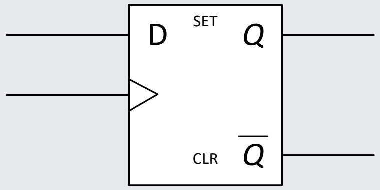
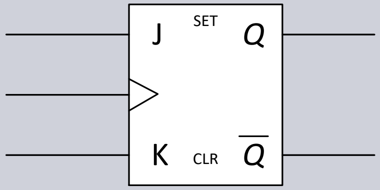
- Same as S-R flip-flop, except that it allows both inputs to be 1, which will toggle the state.



J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

You are encouraged to derive this table yourselves.

Summary

Flip-flop	Graphical symbol	Truth table
S-R		S / R / Q_{n+1}: 0 0 Q_n; 0 1 0; 1 0 1; 1 1 -
D		D / Q_{n+1}: 0 0; 1 1
J-K		J / K / Q_{n+1}: 0 0 Q_n; 0 1 0; 1 0 1; 1 1 $\overline{Q_n}$

S-R truth table:

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	-

D truth table:

D	Q_{n+1}
0	0
1	1

J-K truth table:

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

Usage example

- Suppose we want to design a 2-bit decremental counter J-K flip flops that repeatedly produces the sequence of output of 3, 2, 1, 0, 3, 2, 1, 0, …
- Number of flip-flops needed : 2

Design

Current		Next		Inputs			
Q_B	Q_A	Q_B	Q_A	J_B	K_B	J_A	K_A
0	0	1	1	1	d	1	d
0	1	0	0	0	d	d	1
1	0	0	1	d	1	1	d
1	1	1	0	d	0	d	1

$$\begin{array}{c|cc} J_B & Q_A{=}0 & 1 \\ \hline Q_B{=}0 & 1 & \\ 1 & d & d \end{array}$$

$J_B = \overline{Q_A}$

$$\begin{array}{c|cc} K_B & Q_A{=}0 & 1 \\ \hline Q_B{=}0 & d & d \\ 1 & 1 & \end{array}$$

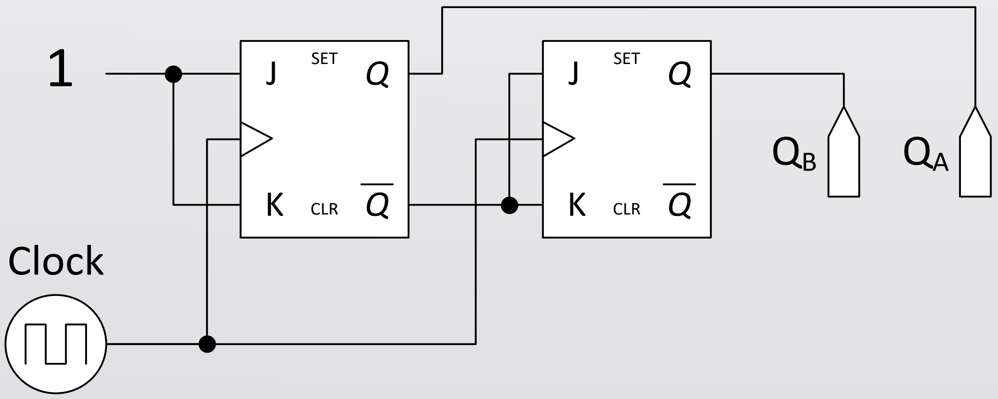
$K_B = \overline{Q_A}$

$$\begin{array}{c|cc} J_A & Q_A{=}0 & 1 \\ \hline Q_B{=}0 & 1 & d \\ 1 & 1 & d \end{array}$$

$J_A = 1$

$$\begin{array}{c|cc} K_A & Q_A{=}0 & 1 \\ \hline Q_B{=}0 & d & 1 \\ 1 & d & 1 \end{array}$$

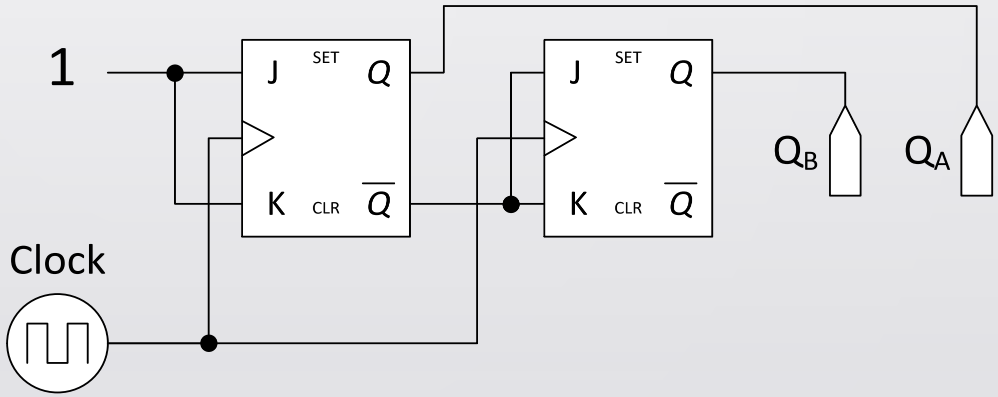
$K_A = 1$



Alternative way

Current		Next	
Q_B	Q_A	Q_B	Q_A
0	0	1	1
0	1	0	0
1	0	0	1
1	1	1	0

Observation
Q_A is simply toggling each time
Q_B is toggled only when Q_A is 0

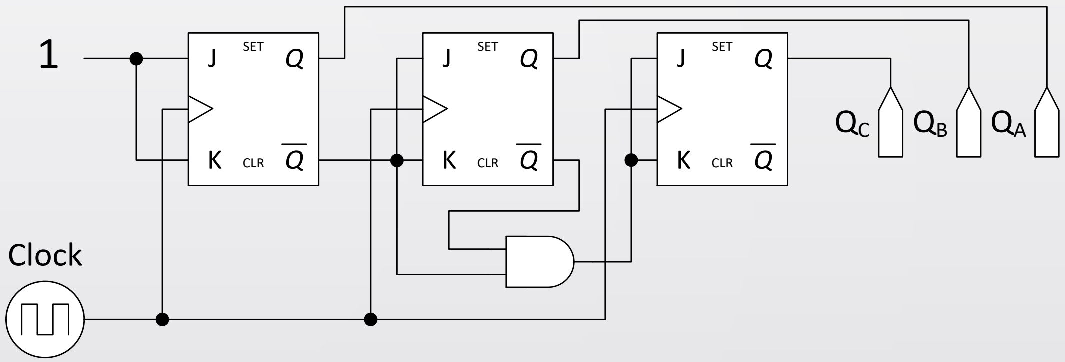


Exercise

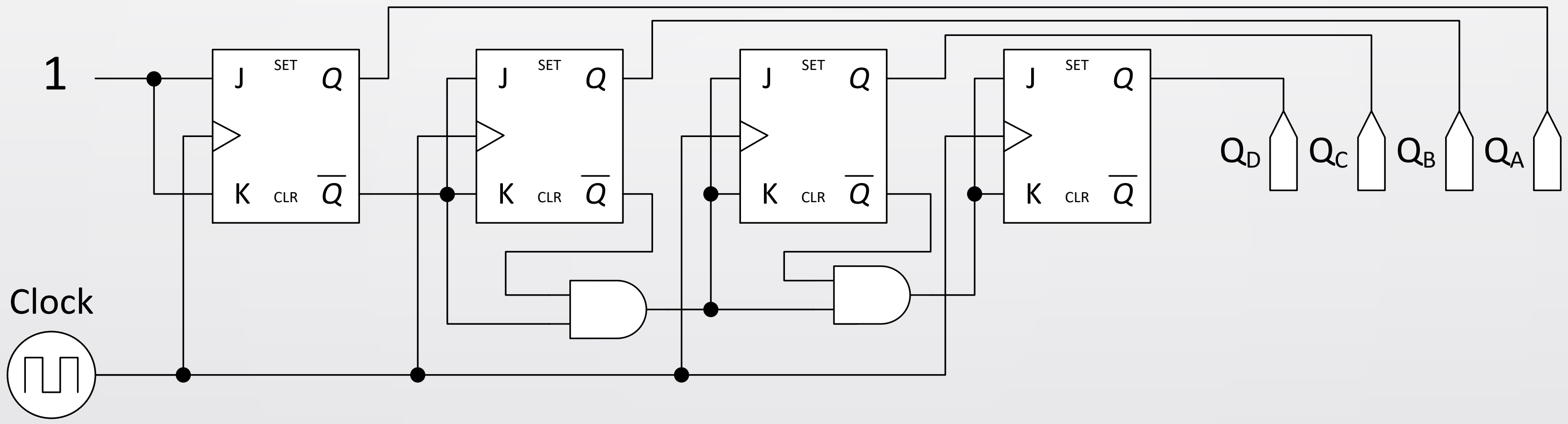
- Expand the previous example to a 3-bit decremental counter.
    - How about 4-bit?

Solution

Current			Next		
Q_C	Q_B	Q_A	Q_C	Q_B	Q_A
0	0	0	1	1	1
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0



Solution (4-bit)


1
J
SET
Q
K
CLR
$\overline{Q}$
Clock
Q_D
Q_C
Q_B
Q_A